SCARA Robot Path Planning with Collision Avoidance using a Radial Basis Probabilistic Roadmap ⋆

Emerson V. A. Dias * **Josias G. Batista** * **Catarina G. B. P. Silva** *
Geraldo L. B. Ramalho * **Darielson A. Souza** **
José Leonardo N. Silva * **André P. Moreira** ??

* *Mobile Robotics Laboratory - Department of Industry, Federal Institute of Education, Science and Technology of Ceará - IFCE, Campus Fortaleza, Fortaleza, CE, Brazil.(e-mail: emersonverasifce@gmail.com, josiasbatista@ifce.edu.br, catarinagbezerra@gmail.com, gramalho@ifce.edu.br, leonardo.silva@ifce.edu.br, apmoreira@ifce.edu.br).*
** *Research Group on Automation, Control and Robotics - Department of Electrical Engineering, Federal University of Ceará, Fortaleza, CE, Brazil (e-mail: darielson@dee.ufc.br)*

Abstract: This paper proposes a Path Planning with Collision Avoidance based on a Radial Basis Function (RBF) trained with random points generated by a Probabilistic Roadmap algorithm. Experiments were performed on a computational model of a SCARA manipulator. The trajectory achieved was evaluated using computational cost, R^2 (multiple correlation coefficient) and root mean square error (RMSE). The results of the trajectory generated by the algorithms in the Cartesian space and also the trajectories of each joint of the manipulator, calculated from the inverse kinematics, show that RBF proved to be an efficient path estimator. The result was compared with Artificial Neural Networks Multilayer Perceptron (MLP) algorithm, where the RBF proved to be more efficient.

Keywords: Radial basis function, probabilistic roadmap, collision avoidance, path planning, SCARA robot.

1. INTRODUCTION

Over the years, following the development of the industrial revolution, robotics started to be inserted in the processes of this sector in order to provide a productive gain. With technological evolution, the use of robotics in the industrial environment has increased and, therefore, several types of robots were created with the purpose of helping or even replacing man in certain tasks (Dias et al., 2021; Souza, 2008). In this way, the most diverse activities within the industry started to be carried out by robots, and it is necessary that they achieve their functions effectively and safely.

In the industry, the execution of tasks by robots occurs in classified areas, which may or may not involve the construction of delimited areas to promote the separation between machine and operators Batista et al. (2017). Despite the growth in efficiency, there is still a difficulty in interaction between operator and machine. The industry aims at human-machine interaction, that is, man and machine working together (Dias et al., 2021; Guerin et al., 2019). However, the possibility of accidents at factories, such as collisions, cannot be ruled out.

According to a study by Pedro (2013) and Haddadin et al. (2008), in cases of impact with a human being, the speed of the robot has a more significant role than its mass, and can put the life of an operator at risk. Therefore, it is essential that the robot is able to complete its activity, avoiding obstacles and collisions with the human being.

In order to avoid these collisions and trace the best possible trajectory, collision prevention algorithms are used. PRM is a powerful method for generating collision-free paths for manipulators with high degrees of freedom (DOF) in non-dynamic environments (Safeea, 2020; Kavraki et al., 1996). The method consists of generating random configurations in the space where the manipulator is located. However, as PRM is entirely based on random sampling, the search path is random, so it is possible that the eventual search path is not always the best path (Gang and Wang, 2016). To work around this problem, there are ways to optimize the algorithm.

In order to improve the result found with PRM, we can use computational intelligence techniques such as Artificial Neural Networks (ANN). Artificial neural networks were developed as generalizations of mathematical models of the biological nervous system (Abraham, 2005).

In this paper, the Probabilistic Roadmap (PRM) algorithm provides input data to generate an estimator based

⋆ This work was supported by the program PIBIC/PIBIT 2021/2022 funded by the IFCE/CNPq/FUNCAP.

on Radial Basis Function (RBF) and Multilayer Perceptron (MLP) Artificial Neural Network (ANN).This paper aims to implement the artificial neural networks MLP and evaluate the performance of the RBF ANN trained with data from a probabilistic roadmap algorithm applied to a SCARA (Selective Compliance Assembly Robot Arm) manipulator in the generation of collision avoidance trajectories in a scene with static obstacles. The implementation of the method is demonstrated from two obstacles, in which the algorithm is submitted to the challenge procedure of identifying the obstacles and elaborating a safe trajectory from the starting point ($q_{initial}$) to the final point (q_{final}). The result also shows the computational costs, R^2 (multiple correlation coefficient) and root mean square error (RMSE).

The main contribution of this work is based on the implementation of the RBF and MLP ANN with PRM for collision avoidance trajectory generation. The ANN algorithm learns the PRM so that the trajectories generated by the PRM are improved of form optimally and without collision, thereby spending less time to be executed. Other contributions can be mentioned: optimized PRM with RNA algorithm for collision avoidance trajectory generation; application of the PRM algorithm to a SCARA manipulator; PRM with RBF and MLP ANN algorithm for collision avoidance trajectory generation.

The rest of the article breaks down as follows. Section 2 describes the characteristics of the SCARA manipulator and the forward and inverse kinematics model. Section 3 presents the methodology used, as well as the description of the PRM, MLP and RBF algorithms. The results are presented in Section 4 and finally the conclusions are mentioned in Section 5.

2. SCARA MANIPULATOR

The SCARA (Selective Compliance Assembly Robot Arm) manipulator is a 3-DOF robot shown in Figure 1. In this work, we use only 2 DOF. The first two joints revolve around the vertical axis (z_1 and z_2) performing together parallel to the horizontal plane X_E Y_E, thus behaving as a 2-DOF planar robot (see Figure 2).

Figure 1. Robotic manipulator SCARA.

2.1 Forward Kinematics

We use the Denavit-Hartenberg (DH) convention to develop the kinematic model and compute the parameters α, a, d and θ using the manipulator coordinate system presented in Figure 2 (Hartenberg and Danavit, 1964).

To find the direct kinematic model, it is used the Denavit-Hartenberg (DH) convention. So, in order to calculate the parameters α, a , d and θ the manipulator coordinate system, presented in Figure 2, was adopted DH convention (Hartenberg and Danavit, 1964).

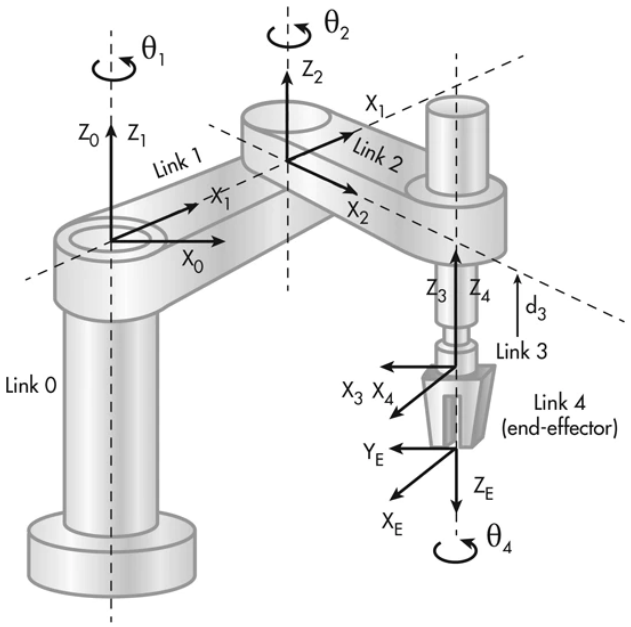


Figure 2. SCARA robot joint coordinate systems.

Table 1. DH Parameters of the SCARA manipulator.

Link	a_i	α_i	d_i	θ_i
1	0.35	0	0.32	θ_1
2	0.30	π	0	θ_2
3	0	0	d_3	0

The homogeneous transformation matrices of the coordinate systems, composed of four basic transformations, are found from the DH parameters. The positions in the workspace can be achieved from the joints space coordinates, as observed in equations (Romano and Dutra, 2002):

$$P_x = 0.35cos(\theta_1) + 0.30cos(\theta_1 + \theta_2) \quad (1)$$

$$P_y = 0.35sen(\theta_1) + 0.30sen(\theta_1 + \theta_2) \quad (2)$$

The equations (1) and (2) are the direct position kinematics problem solution for the SCARA manipulator.

2.2 Inverse Kinematics of SCARA

From the equations of direct kinematics (1) and (2) and applying some trigonometric transformations we find the inverse kinematics equations, given by (Batista et al., 2020):

$$\theta_1 = tan^{-1}\left[\frac{P_y(L_1 + L_2cos(\theta_2)) - P_xL_2sen(\theta_2)}{P_x(L_1 + L_2cos(\theta_2)) - P_yL_2sen(\theta_2}\right] \quad (3)$$

$$\theta_2 = cos^{-1}\left(\frac{P_x^2 + P_y^2 - L_1^2 - L_2^2}{2L_1L_2}\right) \quad (4)$$

where $L_1 = 0.35$ m and $L_2 = 0.30$ m, are the length values of each manipulator joint, as shown in Figure 2.

3. METHODOLOGY

3.1 Probabilistic Roadmap Method

Probabilistic Roadmap (PRM) method provides motion planning to find a collision avoidance path. It is successfully used in mobile robots in the presence of obstacles (Mohanta and Keshari, 2019). This methodology is extremely efficient due to the ability to design paths quickly and the prevalence of the shortest path. In this method, a random sample of the configuration space is initially generated using a uniform probability distribution. In sequence, the algorithm tests the sample for collision, if it is not detected, the trajectory of the point will be generated $q_{initial}$ to q_{final} (Sciavicco et al., 2011; Dias et al., 2021; Siciliano et al., 2009).

PRM randomly generates a set of configurations, which are represented as nodes. Then the planner connects these nodes until a more efficient path is elaborated (Spong et al., 2020). In the literature directed to PRM, these are the main functions used to determine the best connection between the created nodes, from the calculated distance:

$$||q' - q|| = [\sum_{i=1}^{n}(q'_i - q_i)^2]^{\frac{1}{2}}, \tag{5}$$

$$max_n|q'_i - q_i|, \tag{6}$$

$$[\sum p\epsilon A||p(q') - p(q)||^2]^{\frac{1}{2}}, \tag{7}$$

$$max_p\epsilon A||p(q') - p(q)||. \tag{8}$$

The logic of the Probabilistic Roadmap (PRM) is based on previously analyzing the robot's trajectory from a predetermined map. Therefore, this knowledge of the location that the robot will transit is possible for it to calculate the route it will follow. Therefore, the SCARA manipulator will calculate the route effectively, aiming to distance all obstacles along the path.

In the PRM, the map described for the machine will be analyzed, and verified, so that it is identified where obstacles and open access roads are. The method is based on random plotting of imaginary points. These points will be fixed in places that were determined to be free, that is, without obstacles. The trajectory developed by this algorithm values not only collision avoidance movement, but also the shortest path Sciavicco et al. (2011).

For trajectory planning with PRM, the following steps are necessary:

(1) The path is a graph $G(V, E)$;
(2) The robot configuration $q \rightarrow Q_{free}$ is a vertex;
(3) The edge $(q_1,\ q_2)$ implies a collision avoidance path between these robot configurations;
(4) A metric is required to $d(q_1,\ q_2)$ (for example, euclidean distance);
(5) Use of coarse knot sampling and fine edge;
(6) Result: a path in Q_{free}.

The pseudo-code of the Probabilistic Roadmap Algorithm 1 presented below.

Algorithm 1. Algorithm Probabilistic Roadmap.
Input:
n: number of input nodes in the roadmap
k: number of neighbors for each configuration
Output:
A roadmap $G = (V, E)$

$V \leftarrow \emptyset$
$E \leftarrow \emptyset$
while $|V| < n$ **do**
repeat
$q \leftarrow a$ random configuration in Q
until when q is collision free
$V \leftarrow V \cup \{q\}$
end while
for all $q \in V$ **do**
$N_q \leftarrow k$ neighbor's q chosen from V according to the distance
for all $q' \in N_q$ **do**
if $(q,\ q') \notin E$ **and** $\Delta(q,\ q') \neq$ null **then**
$E \leftarrow E \cup \{(q,\ q')\}$
end if
end for
end for

3.2 Multilayer Perceptron (MLP) ANN

One of the benefits of using Artificial Neural Networks as a method to solve the problem stated is the high possibility of obtaining linear and nonlinear models. The Multilayer Perceptron Artificial Neural Network (MLP ANN) is the most commonly used type to solve multiclass or nonlinear problems. This type of ANN's is a supervised learning network since it requires not only input but also output data to perform learning of a mapping between input and output variables (Rocha et al., 2021). The MLP network has the main function of creating a model that correlates the inputs and outputs of a system. Figure 3 shows an MLP ANN.

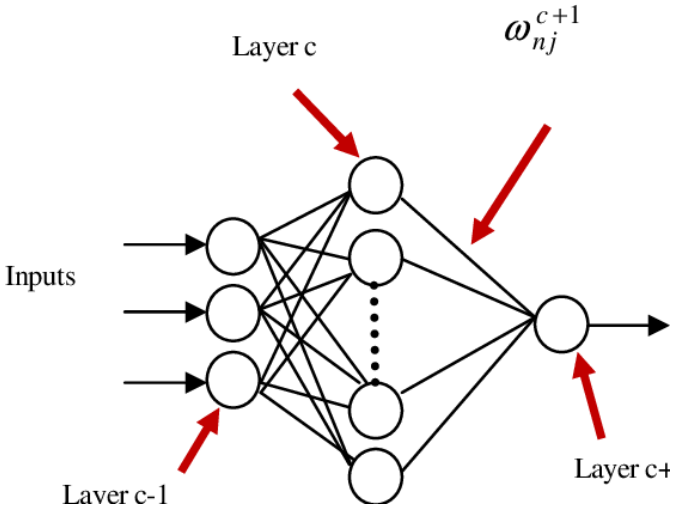


Figure 3. Basic structure of the MLP ANN.

MLP ANN can learn from an algorithm called backpropagation. Therefore, an iteration of the input data in the

neural network is made, where in each loop the actual output of the network is compared with the deed, thus an error is obtained. The goal is to minimize this error by adjusting the weights until a desired convergence is achieved.

Algorithm 2. MLP ANN Training.
1: Generate Wa_{ji} randomly the weights and bias B_i, where i = 1, ..., N of the N neurons;
2: Calculate the output of the hidden layer;
3: Calculate the output weights;
4: Error calculation;
5: Backpropagation Training.

The backpropagation algorithm is used in the training of multilayer neural networks with one or more hidden layers. A brief summary of MLP ANN retropropagation training is presented. The neuron error response j in loop n is represented by:

$$e_j(n) = s_j(n) - y_i(n), \tag{9}$$

where $s_j(n)$ is the desired response for neuron j of the output layer. The instantaneous value of the quadratic error for neuron j is defined by $\frac{1}{2}e_j^2(n)$. The sum of the instantaneous quadratic errors of the network is then defined by:

$$\epsilon(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n), \tag{10}$$

where C is the set that contains all the neurons of the network output layer. Let N be the total number of training patterns contained in the training set. The mean square error is then defined by:

$$\epsilon_{av} = \frac{1}{N}\sum_{n=1}^{N}\epsilon(n) \tag{11}$$

The Algorithm 2 show training process where the output neuron (y = j) fed by the activation's of all the neurons of the immediately preceding layer. The activation function used was the logistic sigmoid. The level of internal activation of neuron j is given by:

$$v_j(n) = \sum_{i=0}^{P} w_{ji}y_i(n), \tag{12}$$

where the variable j the amount of neurons in the hidden layer P represents the number of entries without the bias. Therefore the activation y_j(n) of neuron j is given by:

$$y_j(n) = f_j(v_j(n)), \tag{13}$$

To minimize the mean square error first need to determine the instantaneous gradient $\frac{\partial\epsilon(n)}{\partial w_{ji}(n)}$. Applying the chain rule one can express this gradient as (Rocha et al., 2021):

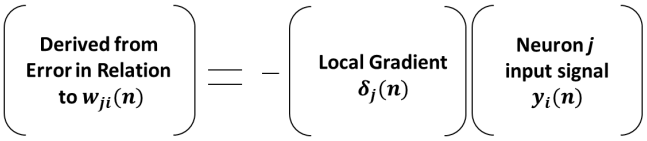
$$\frac{\partial\epsilon(n)}{\partial w_{ij}(n)} = \frac{\partial\epsilon(n)}{\partial e_i(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial yj(n)}{\partial v_j(n)}\frac{\partial vj(n)}{\partial w_{ji}(n)} \tag{14}$$

The adjustment rule is given by (Rocha et al., 2021):

$$w_{ji}(n+1) = w_{ji}(n) - \eta(n)\frac{\partial\epsilon(n)}{\partial w_{ji}(n)}, \tag{15}$$

where $\eta(n)$ is the learning rate in iteration n.

The backpropagation training algorithm can be summarized as: the partial derivative of the sum of the instantaneous errors with respect to the weight $w_{ji}(n)$ that connects neuron i to neuron j which can be shown below:



The calculation of the local gradient $\eta_j(n)$ depends on whether the neuron is an output or hidden neuron. The parameters chosen for the MLP ANN configuration is 1 hidden layer, 50 neurons in each hidden layer, 1000 epochs and 0.001 learning rate.

3.3 Radial Basis Function (RBF) ANN

Figure 4 presents the architecture of an RBF neural network that is composed of 3 layers, input layer, hidden layer and output layer. According to (Souza et al., 2019) the hidden layer operation is done by a nonlinear transformation of the input layer data.

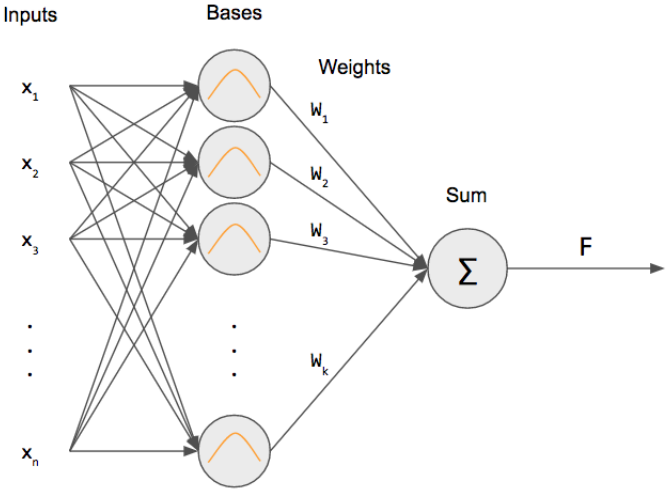


Figure 4. Architecture of RBF ANN.

Let an integer P<N (N = number of data of training). The Equation (16) provides the output of the neural pattern of Figure 4:

$$\sum_{k=0}^{P} w_k\varphi(x;t_k) + w_o, \tag{16}$$

where t_k represents the centers of radial basis functions.

Using regularization in the training of this neural network the function ϵ_R to be minimized is described by:

$$\epsilon_R = \sum_{i=1}^{N}(d_i - f(x_i))^2 + \sum_{j=1}^{P}\lambda_j w^2 j, \tag{17}$$

Where the desired output vector is given by d = [d_1, d_2, ..., d_N]^T, the synaptic weight vector is given by w = [w_0,w_1, w_2, ..., w_p]^T, the neural network response is given by $f(x_i)$ and the regularization term is given by λ_j.

During training, the locations of the centers t_k of the radial basis functions are found. The radial basis function chosen for the work was Gaussian as defined by:

$$\varphi(x, t_k) = \exp\left(-\frac{1}{\sigma_k^2}||x - t_k||^2\right),\ 6k = 1, 2, 3, .., P \quad (18)$$

where σ represents the width of the radial function and t_k its center. The output of the RBF ANN is given by:

$$y = \sum_{k=0}^{P} w_k \exp\left(-\frac{1}{\sigma_k^2}||x - t_k||^2\right) + w_o \quad (19)$$

The Φ wrappings where is the representation of the interpolation matrix.

$$\Phi = \begin{bmatrix} 1 & \varphi(x_1, t_1) & \varphi(x_1, t_2) & . & \varphi(x_1, t_p) \\ 1 & \varphi(x_2, t_1) & \varphi(x_2, t_2) & . & \varphi(x_2, t_p) \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & \varphi(x_N, t_1) & \varphi(x_N, t_2) & . & \varphi(x_N, t_P) \end{bmatrix}, \quad (20)$$

The Equation (20) can be described as:

$$w = (\Phi^T.\Phi + Q)^{-1}\Phi.d \quad (21)$$

where Q is:

$$Q = \begin{bmatrix} \lambda_1 & 0 & . & 0 \\ 0 & \lambda_2 & . & 0 \\ . & . & . & . \\ 0 & 0 & . & \lambda_r \end{bmatrix}, \quad (22)$$

For the training of the centers of the RBF network it was used the kmeans unsupervised learning method. The kmeans algorithm was proposed used in clustering applications. Let $c(t)_{j=1}^{m}$ be the centers of the radial basis functions in iteration t. The kmeans algorithm can be described as follows:

Algorithm 3. Training centers with kmeans.
1: Choice of distinct random values for the centers $c_j(t)$;
2: Take a random vector x_i from the set of input patterns;
3: Determine the index k of the center closest to the input pattern as: $k(x_i) = \arg\min_j||x_j(t) - c_j(t)||$, $j = 1...m$;
4: Adjust the centers using the following rule of the Equation (23), where $\gamma \in (0,1)$ is the adjustment rate;
5: Repeat steps 2 to 5 for all N input patterns and until the centers do not show significant change after each complete presentation of the N patterns.

$$\begin{cases} c_j(t+1) = c_j(t) + \gamma[x_i(t) - c_j(t)], k = k(x_i) \\ c_j(t), \end{cases} \quad (23)$$

3.4 Initial conditions

The Probabilistic Roadmap method is implemented on a map that has two circular obstacles. The scenario is formed by the initial position (0.4; 0.7), final position (0.4; 0.1) of the manipulator, in Cartesian space. For obstacles, the positions (0.3; 0.4) were used for obstacle 1 (which is more to the left), and (0.5; 0.4) for the obstacle 2 (which is more to the right) in the scene, positions that define their center in the Cartesian space.

Figure 5 represents the application of the Probabilistic Road map, which based on random points, elaborates a collision-free trajectory.

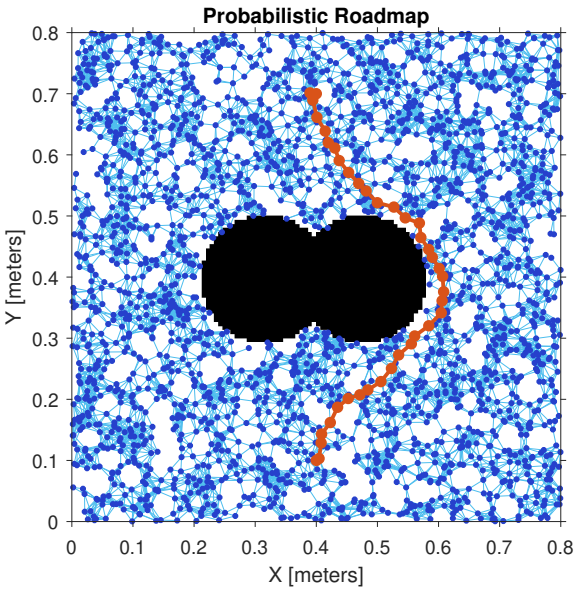


Figure 5. Trajectory elaborated by PRM.

The algorithm elaborates a collision-free trajectory based on random points in Cartesian space. Based on the points chosen by the PRM, these will be used as input data for the MLP and RBF networks. In addition, the inverse kinematics for the trajectory generated by the PRM will be calculated, resulting in the output that will be used to train the neural networks. Thus, using the acquired data, the networks will be trained to elaborate a collision-free trajectory based on the PRM method.

In the present study, the architecture of ANN MLP and RBF based on PRM was used to learn or collision-free path to generate collision-free costumes of the SCARA manipulator. Furthermore, the parameters chosen for RBF ANN configuration was 1 hidden layer, 10 centers and 1000 epochs, for MLP ANN the parameters chosen was 50 hidden layers and 1000 epochs.

3.5 Evaluation metrics

All methods were evaluated by root mean square error (RMSE) in Equation (24) and multiple correlation coefficient (R^2) in Equation (25).

$$RMSE = \frac{\sum_{t=1}^{T}(\hat{y}_{(t)} - y_{(t)})^2}{T} \quad (24)$$

where $\hat{y}$ is the prediction, y is the observed variable and T is the number of data points used for testing.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_{(i)} - \hat{y}_{(i)})^2}{\sum_{i=1}^{n}(y_{(i)} - \bar{y}_{(i)})^2} \quad (25)$$

where:
$y_{(i)}$: is observation;
$\hat{y}_{(i)}$: is prediction;
$\bar{y}_{(i)}$: is average of observations.

4. RESULTS AND DISCUSSIONS

To generate a trajectory in cartesian space using PRM, 2500 points were used with a distance of 0.03 between

them. These points chosen by PRM to form the collision-free path are used as input data for MLP and RBF networks.

4.1 MLP Results

Figure 6 presents the real and estimated path generated by MLP based on PRM collision free.

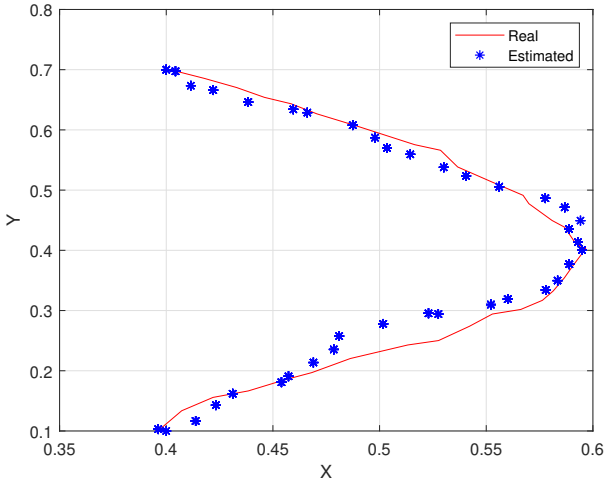


Figure 6. Trajectory generated by MLP based on PRM.

Figure 7 show the trajectory of joints 1 and 2 of the manipulator for the collision-free path of the MLP network found from the inverse kinematics model (equations (3) and (4)) and from the points found in Figure 6.

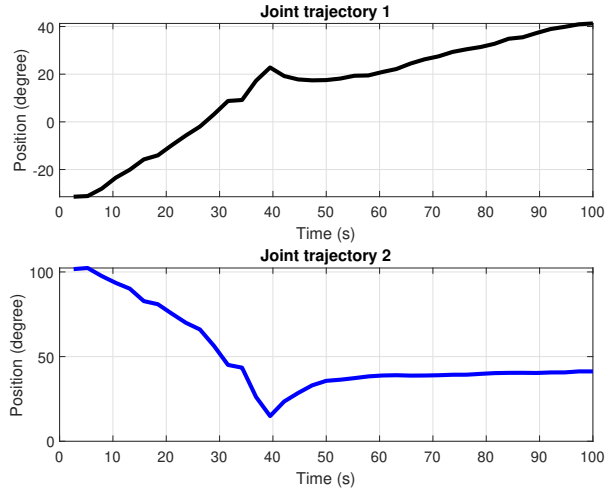


Figure 7. Joints trajectory generated by MLP based on PRM.

4.2 RBF Results

Figure 8 presents the real and estimated trajectories generated by RBF based on PRM.

Figure 9 show the trajectory of joints 1 and 2 of the manipulator for the collision-free path of the MLP network found from the inverse kinematics model (Equations (3) and (4)) and from the points found in Figure 8.

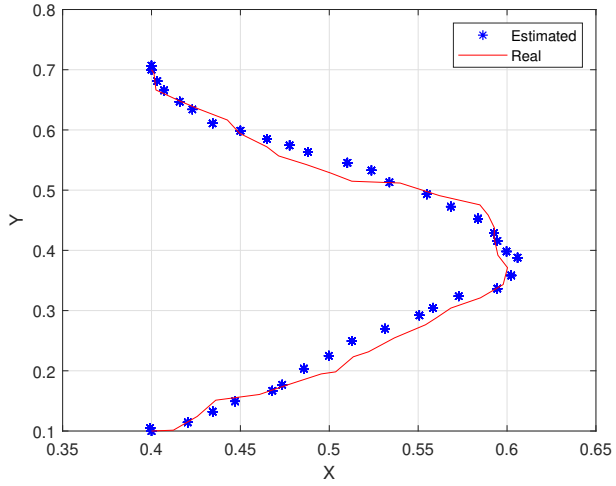


Figure 8. Trajectory generated by RBF based on PRM.

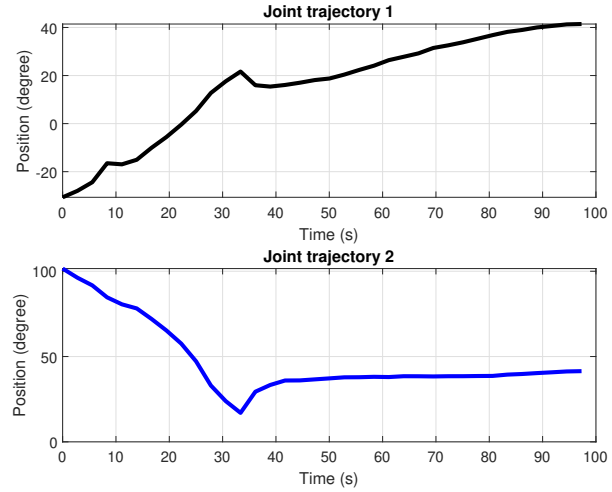


Figure 9. Joints trajectory generated by RBF based on PRM.

4.3 Discussions

A comparison was made between the algorithms of the two ANN. The Root Mean Square Error, R^2, number of neurons and computational cost for each ANN were analyzed.

Table 2. Comparison of algorithms through evaluation metrics.

ANN	RMSE	R^2	Comp. coast [s]
MLP	3.2 e-09	0.9999	12.7432
RBF	9.7834 e-05	1.0000	0.200434

Table 3 presents the epochs number, hidden layer number, and number of neurons in the hidden layer (h-l) and in the output layer (o-l).

Observing the source data in Table 2, it is possible to see that in terms of RMSE, for the same number of neurons in the hidden layer, the MLP network has the lowest value. However, observing the computational cost, it can be seen that the RBF network has a higher speed than the MLP

Table 3. Comparison of ANN algorithm parameters.

ANN	Epoch	Hidden layer	Neurons h-l	Neurons o-l
MLP	1000	1	50	1
RBF	1	1	50	2

network, since its execution time was considerably lower. The computational cost is important when it comes to real-time implementation, as it is this time that informs if it is possible for the algorithm to do the collision avoidance with the manipulator in motion.

5. CONCLUSIONS

The MLP and RBF algorithms, based on PRM, are efficient in applications where a collision-free trajectory is required, and it can be afirmated that this study has a contribution in this context. This article compared the use of two collision-free path algorithms in a SCARA manipulator. Based on the results of trajectory elaboration by random points of the PRM, both neural networks obtained satisfactory results.

In carrying out the trajectory from the data generated by the PRM, the variable that stood out in the difference between the MLP and the RBF was the computational cost, for the MLP the time was 12.7432 s, while for the RBF it was 0.2004s. For the RMSE, the MLP had a low value of 3.2 e-09, while the RBF had a value of 9.7834 e-05. For R^2 the difference was minimal. Therefore, both algorithms are satisfactory in collision avoidance for the manipulator under study. However, the RBF network presented a much better computational cost.

As future work we intend to: Implement algorithms that develop collision-free trajectories, for the SCARA manipulator, based on computer vision.

REFERENCES

Abraham, A. (2005). Artificial neural networks. *Handbook of measuring system design*.

Batista, J., Souza, D., Dos Reis, L., Barbosa, A., and Araújo, R. (2020). Dynamic model and inverse kinematic identification of a 3-dof manipulator using rlspso. *Sensors*, 20(2), 416.

Batista, J.G. et al. (2017). Um estudo de caso sobre a interação máquina-máquina na perspectiva dos algoritmos clássicos de geração de caminhos livres de colisão.

Dias, E.V., Silva, C.G., Batista, J.G., Ramalho, G.L., Costa, J.R., Silva, J.L., and Souza, D.A. (2021). Prevenção de colisão de um manipulador scara utilizando campos potenciais artificiais e caminhos probabilísticos. *Brazilian Journal of Development*, 7(1), 11252–11270.

Gang, L. and Wang, J. (2016). Prm path planning optimization algorithm research. *Wseas Transactions on Systems and control*, 11, 81–86.

Guerin, C., Rauffet, P., Chauvin, C., and Martin, E. (2019). Toward production operator 4.0: modelling human-machine cooperation in industry 4.0 with cognitive work analysis. *IFAC-PapersOnLine*, 52(19), 73–78.

Haddadin, S., Albu-Schaffer, A., and Hirzinger, G. (2008). The role of the robot mass and velocity in physical human-robot interaction-part i: Non-constrained blunt impacts. In *2008 IEEE International Conference on Robotics and Automation*, 1331–1338. IEEE.

Hartenberg, R. and Danavit, J. (1964). *Kinematic synthesis of linkages*. New York: McGraw-Hill.

Kavraki, L.E., Svestka, P., Latombe, J.C., and Overmars, M.H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4), 566–580.

Mohanta, J.C. and Keshari, A. (2019). A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation. *Applied Soft Computing*, 79, 391–409.

Pedro, L.M. (2013). *Uma proposta de sistema robótico para manipulação e interação física segura em ambientes não estruturados*. Ph.D. thesis, Universidade de São Paulo.

Rocha, F.V., Iha, K., and Tolosa, T.A.G.d. (2021). Forecasting chemical characteristics of aircraft fuel using artificial neural networks. *Journal of Aerospace Technology and Management*, 13.

Romano, V. and Dutra, M. (2002). Introdução a robótica industrial. *Robótica Industrial: Aplicação na Indústria de Manufatura e de Processo, São Paulo: Edgard Blücher*, 1–19.

Safeea, M. (2020). *Des robots manipulateurs collaboratifs sûrs*. Ph.D. thesis, Paris, HESAM.

Sciavicco, L., Siciliano, B., Villani, L., and Oriolo, G. (2011). Robotics: Modelling, planning and control, ser. advanced textbooks in control and signal processing.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Force control*. Springer.

Souza, D.A., Reis, L.L., Batista, J.G., Costa, J.R., Junior, A., Araújo, J.P., and Braga, A.P. (2019). Nonlinear identification of a robotic arm using machine learning techniques. In *World Conference on Information Systems and Technologies*, 492–501. Springer.

Souza, S. (2008). *Planejamento de trajetória para um robô móvel com duas rodas utilizando um algoritmo A-Estrela modificado*. Ph.D. thesis, Dissertação de mestrado, Programa de Pós Graduação em Engenharia Elétrica, UFRJ/COPPE, Rio de Janeiro.

Spong, M.W., Hutchinson, S., and Vidyasagar, M. (2020). *Robot modeling and control*. John Wiley & Sons.