RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Digital Object Identifier: 10.24054/rcta.v2i46.3583

Inverse kinematics for a 2R planar robot based on a neural network with synthetic data through a model ensemble approach

Cinemática inversa para un robot planar 2R basada en una red neuronal con datos sintéticos a través de un enfoque de ensamble de modelos

Angie Paola Mancipe García[1], **Nicolas David Barrera Fonseca**[1], **MSc. Jorge Eduardo Cote Ballesteros**[1], **MSc. Jhon Edisson Rodríguez Castellanos**[1], **MSc. Fabián Barrera Prieto**[1]

[1] ***Universidad ECCI,*** *Facultad de Ingeniería, Ingeniería Mecatrónica, Bogotá, Cundinamarca, Colombia.*

Correspondence: *fbarrerap@ecci.edu.co*

Received: *february 20, 2025.* ***Accepted:*** *june 25, 2025.* ***Published:*** *july 01, 2025.*

How to cite: *A. P. Mancipe García, N. D. Barrera Fonseca, J. E. Cote Ballesteros, J. E. Rodríguez Castellanos, and F. Barrera Prieto, "Inverse kinematics for a 2R planar robot based on a neural network with synthetic data through a model ensemble approach", RCTA, vol. 2, no. 46, pp. 40–48, Jul. 2025.*
Recovered from https://ojs.unipamplona.edu.co/index.php/rcta/article/view/3583

Abstract: This article presents the development of a neural network-based model, trained with synthetic data, to replace the geometric inverse kinematics of a 2R planar robot. This approach aims to simplify the implementation of kinematics, reducing development time and computational resource usage. The model was created in Google Colab (Python) using TensorFlow/Keras, which facilitated its creation and training. Furthermore, the system integrates real-time image processing to recognize and follow contours, which the robot subsequently traces. A linear and vertical motion was implemented using a rack-and-pinion mechanism, enabling discontinuous tracing between contours of an image. The results, averaging three neural network models, show high accuracy in predicting the angles of the robot's first two joints, with an RMSE of 0.2293 and 0.0739 compared to the geometric inverse kinematics.

Keywords: 2R robot, Raspberry Pi, inverse kinematics, neural networks, robotics.

Resumen: Este artículo presenta el desarrollo de un modelo basado en redes neuronales, entrenado con datos sintéticos, para reemplazar la cinemática inversa geométrica de un robot planar 2R. Este enfoque busca simplificar la implementación de cinemáticas, reduciendo el tiempo de desarrollo y el uso de recursos computacionales. El modelo se creó en Google Colab (Python) utilizando TensorFlow/Keras, facilitando su creación y entrenamiento. Además, el sistema integra procesamiento de imágenes en tiempo real para reconocer y seguir contornos, los cuales el robot traza posteriormente. Se implementó un movimiento lineal y vertical mediante un mecanismo piñón-cremallera, permitiendo el trazo discontinuo entre los contornos de una imagen. Los resultados, promediando tres modelos de red neuronal, muestran alta precisión en la predicción de los ángulos de las dos primeras articulaciones del robot, con un RMSE de 0.2293 y 0.0739 respecto a la cinemática inversa geométrica.

Palabras clave: robot 2R, raspberry pi, visión por computador, redes neuronales, robótica.

1. INTRODUCTION

In recent years, neural networks have empowered various engineering fields, including robotics, by providing innovative solutions to complex problems [1]. Their ability to model highly nonlinear systems and adapt to dynamic environments makes them important tools in a wide variety of cases [2]. One such application is Inverse Kinematics (IK) in serial configuration robots, where neural networks offer a promising alternative to traditional geometric and mathematical methods.

Historically, inverse kinematics has been addressed using mathematical models that require complex equations and robot specific parameters [3], [4], [5], [6]. Although these methods are effective in standard situations, they present significant limitations when trying to adapt to custom configurations or highly dynamic environments, where variations in geometry or operating conditions require constant adjustments. In these cases, traditional methods may be ineffective or slow. Recent research has shown that machine learning models, and specifically neural networks, may be trained to efficiently handle these variabilities, providing fast and accurate solutions without the need to manually recalculate parameters for each robot configuration or condition, justifying their use in robotic applications that require high adaptability and autonomy [7], [8].

With advances in artificial intelligence, neural networks have established themselves as an effective alternative to face the challenges in robotics. Various studies have shown their impact on improving precision, adaptability and flexibility in robotic applications. For example, the article [9] presents a high precision method for palletizing tasks with a 3 DOF SCARA robot using recurrent neural networks. Likewise, in [10] it is demonstrated how these networks can interpret neurological signals to control robotic devices.

Similarly, in [11] a feedforward neural network was used to solve the inverse kinematics of a three-link planar manipulator, achieving accurate trajectories even in complex configurations where traditional analytical methods are not feasible.

Furthermore, in [12] a neural network based controller was developed that improves the tracking of repetitive trajectories by compensating for uncertainties in the robot model. This approach, combined with stability techniques based on the Lyapunov criterion, ensures precise and stable control without the need for constant adjustments.

In the case of 2R planar robots, they stand out for their speed, precision and simplicity, making them ideal tools for tasks such as pattern tracing on flat surfaces, 2D laser cutting, educational experimentation in robotics and simple manipulations that do not require movements outside their work area. [13]. Similarly, approaches such as model assembly allow combining the strengths of various neural networks to improve performance, reducing errors and increasing robustness in prediction against untrained data [14].

In the context of machine learning, one of the most frequent problems is obtaining the data for training and validating the algorithm. Consequently, in recent years the use of synthetic data, obtained from sources other than physical measurements on the robot, has been growing. This work proposes the use of synthetic data for training a neural network to calculate the joint angles of a 2R robot for contour drawing tasks.

The paper is structured in three sections: Section 2 covers the methodology, including the development of the neural network, the kinematics (inverse and forward) and the image processing; Section 3 presents the analysis of the results of the operation of the network in the different working modes and Section 4 presents the conclusions and possible future work.

2. METHODOLOGY

2.1. System Components and their Integration

Fig. 1 presents the interconnection of the physical components that are integrated in this work, where a 2R robot designed in SolidWorks and printed in 3D is observed. In addition, a Raspberry Pi 4B (8 GB of RAM) [15] acts as the controller of the 2R robot, coordinating the interaction between the system components. The PCA9685 board whose control signal is acquired via I2C [16], is responsible for varying the angular position of the three servomotors through independent PWM channels with 12 bits of resolution each. Finally, a 5 MP camera captures images through real time video, for later processing.

The Graphical User Interface (GUI) developed in Qt designer runs on the Raspberry Pi, which allows the execution of two modes of operation: drawing a name of up to nine characters and outlining the contours of an image captured through the camera. The neural network model of inverse kinematics allows the desired trajectory to be carried out. The manipulation and supervision of the robot are carried out remotely, and the image processing is based on previous research in the area. [17], [18], [19].

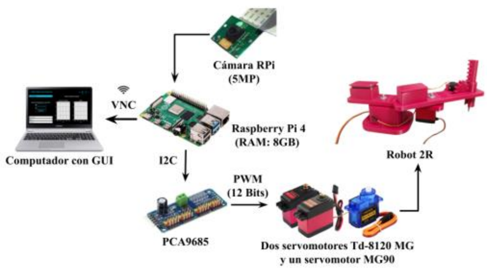


***Fig. 1**. Interconnection of components.*
***Source:** Own elaboration.*

2.2. Multilayer Perceptron Neural Network (MLP)

Multilayer Perceptron (MLP) neural network is a type of neural network composed of several layers (see Fig. 2), an input layer, one or more hidden layers and an output layer; each neuron is connected to those in the next layer, and information is propagated forward using nonlinear activation functions.

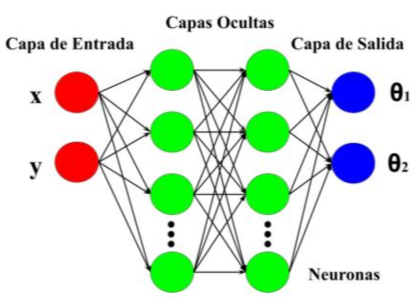


***Fig. 2**. Multilayer Perceptron Neural Network.*
***Source:** Own elaboration.*

The training of the MLP neural network is done using the backpropagation algorithm, which adjusts the weights of the connections to minimize the error between the outputs and the actual values. The output of each neuron in a perceptron is calculated using (1).

$$y = f\left(\sum_{i=1}^{n} x_i w_i + b\right) \quad (1)$$

Where:

- y: Output of the neuron.
- x_i: Inputs of the neuron (with $i = 1, 2, 3, \dots, n$).
- w_i: Weights associated to each input x_i
- b: Bias of the neuron.
- f: Activation function applied to the weighted sum of the inputs.

The development of the neural networks was carried out in Google Colab using the Python language, where a set of 9.872.164 data was created, of which 70% of the data was used for training and 30% for model validation. This data set (see Fig. 3) was obtained from the simulation of the combined movement of the angles θ_1 and θ_2 every 0.001 radians within the workspace, in order to obtain the X and Y positions (inputs of the neural network) with the MTH in (2).

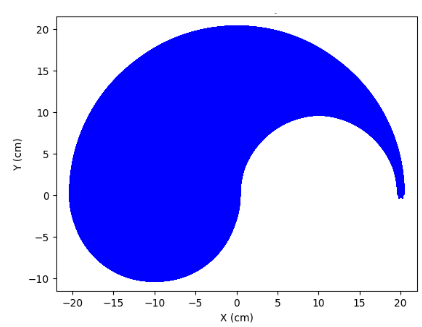


***Fig. 3**. Synthetic data in the 2R robot workspace.*
***Source:** Own elaboration.*

For the implementation of the neural network, the TensorFlow, SciPy and scikit-learn libraries were used. Table 1 shows the most relevant data used in the development of the neural network.

***Table 1:** Neural network training parameters.*

Parameters	Value
Input variables	X and Y
Output variables	θ_1 and θ_2
Training Data	70%
Validation Data	30%
Optimizer	Adam
Activation Function	Sigmoid
Loss Function	MSE
Error Metric	MAE

***Source:** Own elaboration.*

2.3. Kinematics

2.2.1. Forward kinematics

Forward Kinematics (FK) through a Homogeneous Transformation Matrix (HTM) allows to determine the position (translation) and orientation (rotation) of the Tool Center Point (TCP), with respect to the reference system (CS{0}), from the movements of the robot's joints. That is, given a set of angles, the coordinates and Euler angles (Roll, Pitch and Yaw) of the TCP are obtained.

Fig. 4 shows the kinematic diagram of the SCARA robot, with the Coordinate Systems (CS) at each joint, taking into account that the system constants are the link lengths l_1 and l_2 are 10 cm, while l_3 is 2.5 cm; while θ_1 and θ_2 are variables from 0° to 180°, likewise d_3 varies from 0 cm to 2.5 cm. The first two joints are rotational and the third joint is prismatic, which varies the height of the end effector through a rack-and-pinion mechanism.

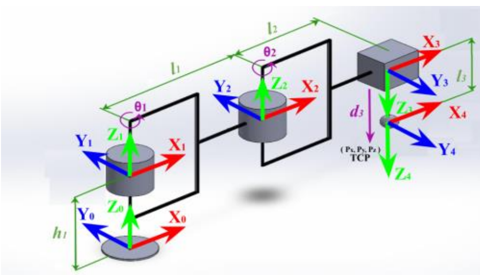


***Fig. 4**. Schematic of the SCARA robot and its coordinate systems.*
***Source:** Own elaboration.*

The Denavit-Hartenberg (DH) method is used, which has a standard and systematic approach to describe the forward kinematics of serial configuration robots, allowing the modeling of joint movements through the assignment of coordinate systems in each joint and in the TCP [20]. In this way, the DH parameters are determined (see Table 2).

***Table 2:** DH method parameters.*

i	θ_i	d_i	α_i	a_i
1	0	h_1	0	0
2	θ_1	0	0	l_1
3	θ_2	0	π	l_2
4	0	$l_3 + d_3$	0	0

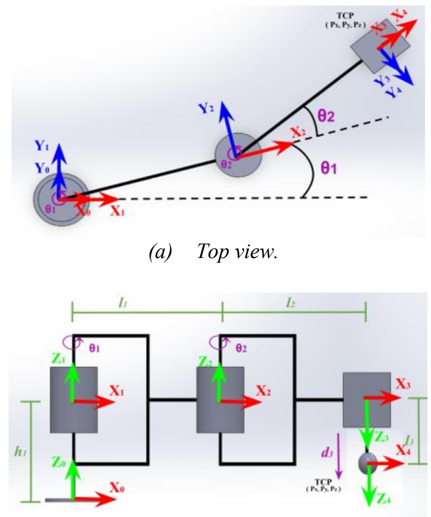
***Source:** Own elaboration.*

The symbolic MTH from CS{0} to CS{4} in (2) represents the position and orientation of the TCP of the robot in Fig. 3, for any value of l_1, l_2, l_3, h_1, θ_1, θ_2 and d_3.

$$T_4^0 = \begin{bmatrix} c(\theta_1+\theta_2) & s(\theta_1+\theta_2) & 0 & l_2\cdot c(\theta_1+\theta_2)+l_1\cdot c(\theta_1) \\ s(\theta_1+\theta_2) & -c(\theta_1+\theta_2) & 0 & l_2\cdot s(\theta_1+\theta_2)+l_1\cdot s(\theta_1) \\ 0 & 0 & -1 & h_1-l_3-d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

2.2.2. Inverse kinematics

Inverse Kinematics (IK) calculates the movements of the robot joints for a desired TCP position and orientation. Fig. 5 shows the views of the SCARA robot, where the projection of the top view (see Fig. 5a) of the rotational joints can be observed, which represent the angles θ_1 and θ_2 in the X, Y plane, and the projection of the side view (see Fig. 5b) shows the linear movement d_3 of the prismatic joint.



(b) Side view.
***Fig. 5**. SCARA robot views*
***Source:** Own elaboration.*

Through a geometric analysis of the top view of the 2R robot in Fig. 5a, (3) and (4) were defined, which allow calculating the angles θ_1 and θ_2 [21]. Similarly, by analyzing the side view of the robot in Fig. 5b, (5) was obtained, which represents the vertical distance d_3, corresponding to the prismatic joint of the SCARA robot.

$$\theta_1 = \tan^{-1}\frac{P_y}{P_x} - \tan^{-1}\frac{l_2\cdot \pm\sqrt{1-\left(\frac{b^2-l_2^2-l_1^2}{2\cdot l_1\cdot l_2}\right)^2}}{l_1+l_2\cdot\frac{b^2-l_2^2-l_1^2}{2\cdot l_1\cdot l_2}} \quad (3)$$

$$\theta_2 = \tan^{-1}\frac{\pm\sqrt{1-\left(\frac{b^2-l_2^2-l_1^2}{2\cdot l_1\cdot l_2}\right)^2}}{\frac{b^2-l_2^2-l_1^2}{2\cdot l_1\cdot l_2}} \quad (4)$$

$$d_3 = h_1 - l_3 - P_z \quad (5)$$

2.4. Image processing

To perform the image processing, two sections were delimited within the robot workspace, as presented in Fig. 6. The blue section is intended for writing alphanumeric characters (maximum 9), while the red section is used to delimit the maximum height and width dimensions of the image to be drawn.

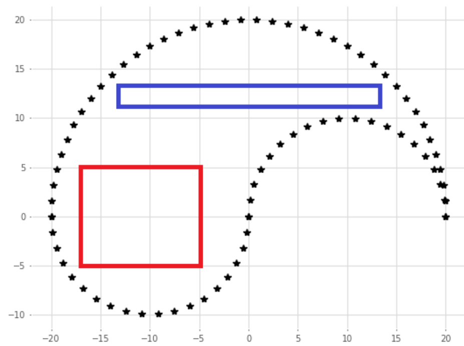
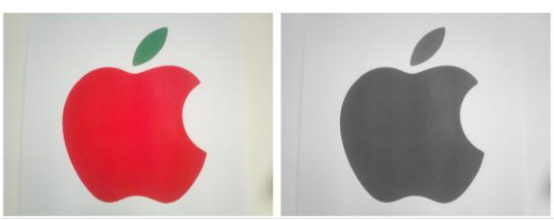


Fig. 6*. Sections for alphanumeric characters and image tracing.*
Source: *Own elaboration.*

From the captured image (e.g. Apple logo (see Fig. 7a)), the image is binarized (see Fig. 7b), that is, it is converted to a black and white scale to simplify the analysis, since in this way only the intensity of the pixels is considered instead of working with multiple color channels. For this process, the binary threshold is set with a minimum value of 117 and a maximum value of 255. Since the image will be taken in real time with a 5 MP camera, it is necessary to set the resolution to 1200 pixels wide to delimit it to the section intended for the contour drawing, keeping the height on a proportional scale to avoid distorting the image.



(a) Original image. *(b) Binarized Apple image.*
Fig. 7*. Apple logo.*
Source: *Own elaboration.*

Subsequently, the image contours are identified, both internal and external, and the coordinates of each of the contours obtained are extracted. In the case of the Apple image, two contours were identified. Finally, each of the coordinates obtained in the previous step is divided by 100 to scale the image and therefore, it is adjusted for the red section of Fig. 6, where the contours will be drawn, which are represented in Fig. 8.

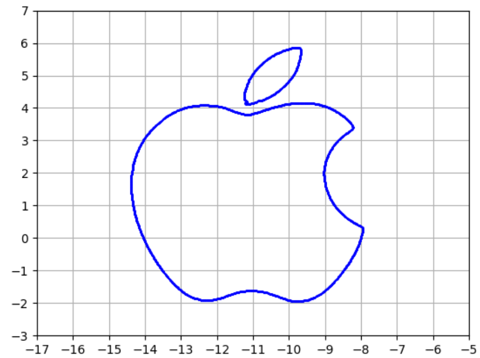


Fig. 8*. Image scaled within the work area section.*
Source: *Own elaboration.*

3. RESULTS

Fig. 9 presents the physical assembly of the SCARA robot, highlighting the integration of the different components of Fig. 1, such as the camera, the Raspberry Pi 4, the computer with the GUI and the PCA9685 module. It is worth noting that, for the latter, a PCB was designed to simplify the connections between devices. The user selects one of the two operation modes through a GUI, which also presents the simulation of the SCARA robot, both modes use the model ensemble neural network approach that replaces the geometric inverse kinematics.

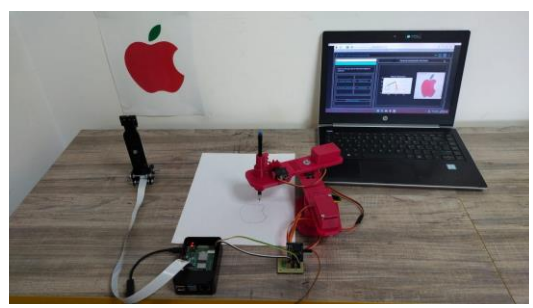


Fig. 9*. Interacción entre componentes y el Robot SCARA.*
Source: *Own elaboration.*

After training the neural network models, an iterative method was implemented that consisted of performing several tests to determine the best combination of parameters taking into account the performance of the models. Table 3 presents the RMSE errors of neural network models for different numbers of neurons, number of hidden layers and epochs, obtaining the ensemble model as the best result.

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

***Table 3:** Parameters of the final model of the Neural Network.*

No.	Model	Neur.	Layers	Ep.	Error (RMSE)
1	Model 1	150	1	90	$\theta_1 = 1.6086$
					$\theta_2 = 0.3717$
2	Model 2	200	2	77	$\theta_1 = 0.4204$
					$\theta_2 = 0.3072$
3	Ensemble model 3.1	200	2	77	$\theta_1 = 0.2293$
	Ensemble model 3.2	210	2	33	$\theta_2 = 0.0739$
	Ensemble model 3.3	250	2	80	

***Source:** Own elaboration.*

Fig. 10 shows the X and Y positions of the Apple logo generated by each of the neural network models compared to geometric inverse kinematics, thereby graphically demonstrating that model 3 is the closest to geometric inverse kinematics.

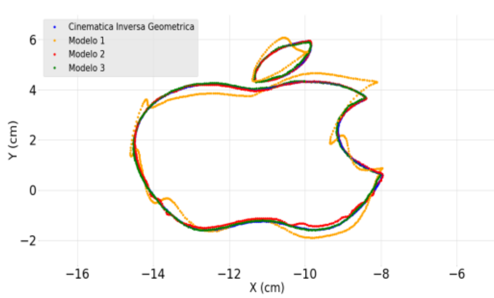


***Fig. 10**. Trajectories of geometric inverse kinematics vs neural network models for the Apple logo.*
***Source:** Own elaboration.*

Fig. 11 presents the comparison of θ_1 and θ_2 calculated by geometric inverse kinematics [22] with the angles (θ_1 and θ_2) obtained with the three neural network models, for each of the 410 points (positions in X and Y) that make up the contours of the Apple logo. Likewise, Fig. 11 shows the behavior of each of the three neural network models with respect to geometric inverse kinematics, where it is evident that the angles (θ_1 and θ_2) of model 3 are the ones that best follow the behavior of θ_1 and θ_2 of geometric inverse kinematics.

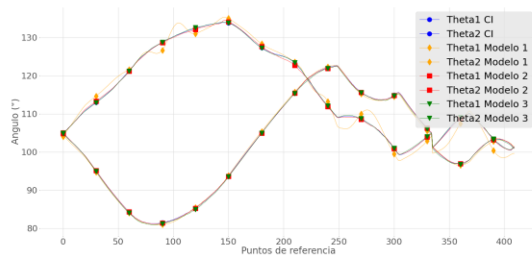


***Fig. 11**. Geometric inverse kinematics vs neural network models for the Apple logo.*
***Source:** Own elaboration.*

Considering Table 3 and Fig. 11, it was determined that the best model of the neural network is model 3, an ensemble model that averages three models. This approach provides a greater generalization capacity, thus outperforming individual models. It was also shown that increasing the number of hidden layers, instead of simply increasing the number of neurons, contributed significantly to improving the performance of the network, since it was possible to reduce the RMSE error. For this reason, this model was selected to be implemented in the estimation of θ_1 and θ_2 in the 2R robot.

3.1. Mode 1: Name writing

Fig. 12 shows the GUI in the writing mode of a string of up to nine characters, in which it is observed that the name entered is "NICOLAS", also showing the simulation of the robot for the trace of said name.

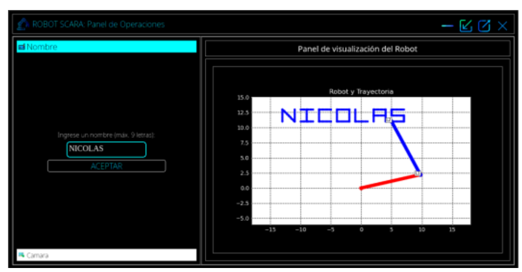


***Fig. 12**. Entering the name and simulation in the GUI.*
***Source:** Own elaboration.*

Fig. 13 shows a significant detail for perform a cleaner and more orderly writing of a string, and it is the discontinuous trace of the name, that is, the spaces between the letters, which is the result of the linear and vertical movement of the prismatic joint that performs the tasks of raising and lowering the pencil in the transitions between characters. This behavior is evidence of the operation of the rack-and-pinion mechanism.

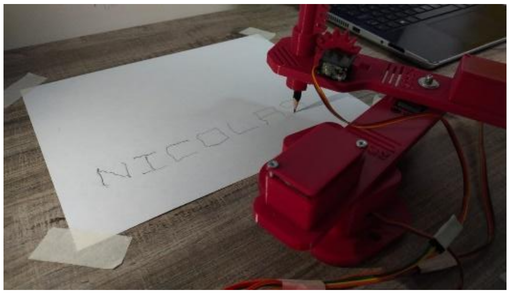


Fig. 13. Writing the name NICOLAS with the physical SCARA robot.
Source: *Own elaboration.*

3.2. Mode 2: Image capture for drawing

Fig. 14 shows the GUI in the camera image capture mode, where the captured image and the identification of the contours of the Apple logo are displayed. In addition, the simulation of the robot's movement in real time is observed.

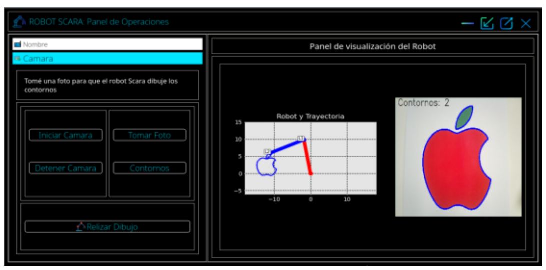


Fig. 14. Image capture and simulation in the GUI.
Source: *Own elaboration.*

Fig. 15 shows the final trace of the Apple logo, physically drawn by the robot. Therefore, the trace of images processed in real time by a camera was demonstrated, which shows flexibility to adapt to new positions within the workspace. However, it is important to highlight that the servomotors play an important role in the contours trace since the resolution of these actuators is not as expected, thus generating an error in the physical angles θ_1 and θ_2 and for this reason the drawn logo has some imperfections in the trace.

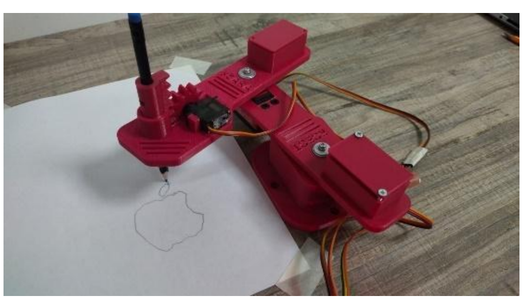


Fig. 15. Tracing the contours of the Apple logo with the physical SCARA robot.
Source: *Own elaboration.*

4. CONCLUSIONS

The results obtained in this project showed that the use of a neural network for the prediction of the inverse kinematics of a 2R robot is an effective alternative because it has low RMSE errors. This method has the advantage for applications with robots of greater complexity and degrees of freedom, where the computational processing of the inverse kinematics increases considerably, as well as the singular positions of said robots.

By implementing the entire system on a Raspberry Pi 4, a faster and more efficient operation of the robot, the camera and the GUI was achieved, since there are no latency times associated with serial communications. In addition, the Raspberry Pi 4 not only manages the GUI and the hardware, but also executes the neural network model, centralizing all operations on a single device. It is important to note that the neural network was trained with synthetic data generated to cover different points within the robot work area, which allowed the model to be trained effectively to solve geometric inverse kinematics tasks with great reliability.

The camera integrated into the system showed satisfactory performance when capturing images from real time video, thereby successfully identifying contours. However, it is essential to consider lighting factors and filters when capturing the image, as these affect the quality of contour identification. In addition, it is essential to adjust the binary threshold parameters to correctly convert the image to black and white; this facilitates contour detection by highlighting areas of interest. It is important to calibrate these threshold values based on lighting conditions, to ensure accurate interpretation of all image details.

With respect to the physical robot trace, some limitations were observed in the precision of the movements, partly due to the performance of the servomotors used. It was noted that linear movements are traced better in the physical robot compared to semicircular movements, which present greater difficulty in maintaining the desired precision and fluidity. To improve the quality and fluidity of the trace, it is advisable to consider replacing servomotors with Direct Current (DC) motors with position control, which would allow greater precision and smoothness in the robot's movements, thus obtaining a final drawing equal to that in the simulation.

REFERENCES

[1] J. L. Sarmiento-Ramos, “Aplicaciones de las redes neuronales y el deep learning a la ingeniería biomédica,” *Revista UIS Ingenierías*, vol. 19, no. 4, 2020, doi: 10.18273/revuin.v19n4-2020001.

[2] J. F. Cortes Zarta, Y. A. Giraldo Tique, and C. F. Vergara Ramirez, “Red neuronal convolucional para la percepción espacial del robot InMoov a través de visión estereoscópica como tecnología de asistencia,” *Enfoque UTE*, vol. 12, no. 4, 2021, doi: 10.29019/enfoqueute.776.

[3] Y. He *et al.*, “Dynamic Modeling, Simulation, and Experimental Verification of a Wafer Handling SCARA Robot with Decoupling Servo Control,” *IEEE Access*, vol. 7, pp. 47143–47153, 2019, doi: 10.1109/ACCESS.2019.2909657.

[4] A. Visioli and G. Legnani, “On the trajectory tracking control of industrial SCARA robot manipulators,” *IEEE Transactions on Industrial Electronics*, vol. 49, no. 1, 2002, doi: 10.1109/41.982266.

[5] K. S. Mohamed Sahari, K. H. Weng, Y. W. Han, A. Anuar, M. Z. Baharuddin, and S. S. K. Mohideen, “Design and development of A 4-dof SCARA robot for educational purposes,” *Jurnal Teknologi (Sciences and Engineering)*, vol. 54, pp. 193–215, 2011, doi: 10.11113/jt.v54.810.

[6] A. Kotani and S. Tellex, “Teaching robots to draw,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2019. doi: 10.1109/ICRA.2019.8793484.

[7] L. H. Ríos González, M. Bueno López, and S. Sánchez A., “Generación de trayectorias para un robot móvil empleando redes neuronal,” *Scientia et Technica*, vol. 2, pp. 94–99, 2008. [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=4742539

[8] J. G. C. Yufra Torres and K. De La Torre Muña, “Diseño de un robot móvil recolector y compactador de botellas de plástico utilizando redes neuronales en playas con arena fina,” *Universidad Ricardo Palma - URP*, 2020, [Online]. Available: https://hdl.handle.net/20.500.14138/3529

[9] J. Hernández Díaz, M. Iparraguirre Monzón, C. Montenegro Castro, J. Alva Alcántara, and E. Manzano Ramos, “MODELADO NO PARAMÉTRICO DE ROBOT SCARA 3-GDL UTILIZANDO REDES NEURONALES RECURRENTES,” *Revista de Investigaciones*, vol. 10, no. 4, pp. 379–390, Dec. 2021, doi: 10.26788/riepg.v10i4.3502.

[10] M. X. González, A. M. Brugiati, D. H. Cornejo, and C. I. Pinzón, “Prototipo de mano robótica controlada mediante el procesamiento de señales cerebrales utilizando redes neuronales recurrentes,” *Revista de Iniciación Científica*, vol. 6, 2021, doi: 10.33412/rev-ric.v6.0.3154.

[11] A.-V. Duka, “Neural Network based Inverse Kinematics Solution for Trajectory Tracking of a Robotic Arm,” *Procedia Technology*, vol. 12, 2014, doi: 10.1016/j.protcy.2013.12.451.

[12] N. Cobanoglu, E. Tatlicioglu, and E. Zergeroglu, “Neural network based repetitive learning control of robot manipulators,” in *Proceedings of the American Control Conference*, 2017. doi: 10.23919/ACC.2017.7963781.

[13] A. De Luca, R. Mattone, and G. Oriolo, “Control of underactuated mechanical systems: Application to the planar 2R robot,” in *Proceedings of the IEEE Conference on Decision and Control*, 1996. doi: 10.1109/cdc.1996.572718.

[14] J. J. Sprockel, A. Fandiño, W. G. Chaves, C. O. Benavides, and J. J. Diaztagle, “Ensamble de redes neuronales artificiales ponderado mediante características operativas para el pronóstico de la insuficiencia cardiaca aguda,” *Revista Colombiana de Cardiología*, vol. 30, no. 5, 2023, doi: 10.24875/rccar.22000065.

[15] Raspberry Pi Ltd, “Raspberry Pi 4 Model B Datasheet,” 2024

[16] NXP Semiconductors, “PCA9685 16-channel, 12-bit PWM Fm+ I2C-bus LED controller,” *NXP Semiconductors N.V*, no. April, 2015.

[17] M. J. Delgado-Gutiérrez, D. F. Herrera-Guillén, L. M. Medina-Barragán, and J. P. Corredor–Gómez, “Implementación de un sistema de procesamiento de imágenes integrado con Raspberry PI 2B para reconocimiento y recolección de fresas maduras,” *Revista Politécnica*, vol. 13, no. 25, 2017, doi: 10.33571/rpolitec.v13n25a6.

[18] G. I. Viera Maza, “Procesamiento de imágenes usando OpenCV aplicado en Raspberry Pi para la clasificación del cacao,” *Thesis*, 2017.

[19] G. Viera, “Procesamiento de imágenes usando OpenCV aplicado en Raspberry Pi

para la clasificación del cacao," *Thesis*, 2017.

[20] C. R. Arroyo Flores and F. J. Triveño Vargas, "Modelamiento y simulación de un robot SCARA de tres grados de libertad empleando SolidWorks y Matlab/Simulink®," *SciELO Analytics*, vol. 10, Mar. 2022, [Online]. Available: http://www.scielo.org.bo/scielo.php?script=sci_arttext&pid=S1683-07892022000100318

[21] S. Kucuk and Z. Bingul, "Robot Kinematics: Forward and Inverse Kinematics," in *Industrial Robotics: Theory, Modelling and Control*, 2006. doi: 10.5772/5015.

[22] A. ; B. N. ; R. D. B. F. Mancipe, "Desarrollo de un brazo robótico de 2 DOF impreso en 3d a través de un pic 18f46k22 y MATLAB para el trazo de contornos de imágenes," *I + T + C Investigación, Tecnología y Ciencia. Vol 1. Num. 17. 2023.*, 2023, [Online]. Available: https://revistas.unicomfacauca.edu.co/ojs/index.php/itc/article/view/406/330