APPLICATION OF NEURAL NETWORKS IN A PARALLEL MANIPULATOR WITH FLEXIBLE LINKS FOR MODEL EXTRACTION

Thiago Liquita Savio[1], Maíra Martins da Silva[2]

[1]*Department of Mechanical Engineering, University of São Paulo*
Av. Trab. São Carlense, 400, 13566-590, São Carlos - SP, Brasil
liquita@usp.br
[2]*Department of Mechanical Engineering, University of São Paulo*
Av. Trab. São Carlense, 400, 13566-590, São Carlos - SP, Brasil
mairams@sc.usp.b

Abstract. Parallel manipulators (PMs) are a viable design alternative for industrial applications. Due to their closed kinematic architecture, they present some advantages compared to their serial counterparts: lightness, high speed/acceleration ratios, high rigidity, load capacity, and high compactness. However, this design option could yield undesired vibrations due to its components' flexibility requiring the implementation of novel joint and task space control strategies. Two main challenges arise when designing a control strategy for a PM: the lack of a direct measurement of the end-effector's pose and their coupling dynamics. This work proposes an estimator for assessing the end-effector's pose of a PM using Artificial Neural Networks using measurements from encoders, strain gauges and camera. The encoders measure the angular displacement of the active joints of the manipulator, the strain gauges the deformation of the links and the camera the position of the end effector. The proposal is validated using experimental data from a 3RRR PM with flexible links. The estimator can be used in control schemes to enhance the performance of flexible PMs.

Keywords: Parallel manipulators; Flexible Links; Neural networks.

1 Introduction

The manipulators are constituted by a set of bodies called links, the links when joined by joints acquire the condition of kinematic chain, that is, kinematic chain can be interpreted as a set of links and joints. A serial kinematic architecture manipulator, it is a manipulator that presents a single kinematic chain, and this is connected to the base. While a parallel kinematic architecture manipulator is one that has more than one kinematic chain connected to the end effector. The parallel kinematic architecture handlers, also known as parallel handlers (PKM). They present several advantages over the traditional architectures of [1] manipulator robots. Among these advantages we can mention lightness, high rigidity, load capacity, high speed/acceleration ratio and high compaction. As a result, this type of architecture has proven to be a great choice when it comes to designing high-performance dynamic and energy-efficient [2] manipulators. However, as is to be expected, making use of these advantages in the design of handlers also has some disadvantages, such as difficulties with the design of control systems of the [3] engine. Taking this difficulty into account, the present work proposes the creation of an estimator to evaluate the pose of the final effector of a PKM with flexible links, using Artificial Neural Networks. Some related works have already been developed. In the work of Bidokhti and Enferadi [4] the problem of the direct kinematics of a robotic planar manipulator 3$\underline{\text{R}}$RR is addressed, this problem is solved using two different methods of Artificial Neural Networks, one is a Multi-Layer Neural Network, while either a Radial-Based Neural Network. Both networks used the resolution of the inverse kinematics of the robot as a training dataset, in the work the effectiveness of the solution is demonstrated by comparing a simulated spiral path with a real path, and for both networks the total error during the simulated path was acceptable, which confirms the reliability of the methods. Elsheikh, Showaib and Asar [5] also studied direct kinematics in 3-RPR, 3-PRR and 3-RRR manipulators with an Artificial Neural Network approach, in his work he claims that the approach proposed by him can certainly learn the input and output data, deducing the nonlinear relationships that are inherent to training. Applications of Artificial Neural Networks are also made to obtain the inverse kinematics of robots. The work of Moori, Khoramdel and Moosavian [6] which is a spherical 3$\underline{\text{R}}$RR parallel robot has as one of its objectives to make quick calculations and overcome the manufacturing uncertainties present in the robot, for that the inverse kinematic equations present in the problem are solved by a Multi-Layer Perceptron Network. The authors Csiszar, Eilers and Verl [7] carried out a work proposing a supervised approach to solve the problem of inverse kinematics and calibration, in this work it was proposed that instead of creating an ideal model for each robot, and then calibrating each one individually, train it through a Neural Network to learn the function of inverse kinematics, since this would already include errors due to manufacturing and/or assembly tolerances.

2 Materials and methods

2.1 Materials

The manipulator under study is the 3($\underline{\text{P}}$)$\underline{\text{R}}$RR (Figure 1), present in the Dynamics laboratory at EESC-USP. This is a closed kinematic chain planar parallel manipulator that has three identical kinematic chains that are linked to the end effector, each of these chains having four joints, one of them prismatic and active ($\underline{\text{P}}$) and the other three ($\underline{\text{R}}$RR) rotational, with the joint furthest from the end-effector being active and the other two passive. For this work, although the manipulator in question can be used as 3($\underline{\text{P}}$)$\underline{\text{R}}$RR, this is not done, since the prismatic joints are not used. So for this work the handler in question can be treated as 3$\underline{\text{R}}$RR.

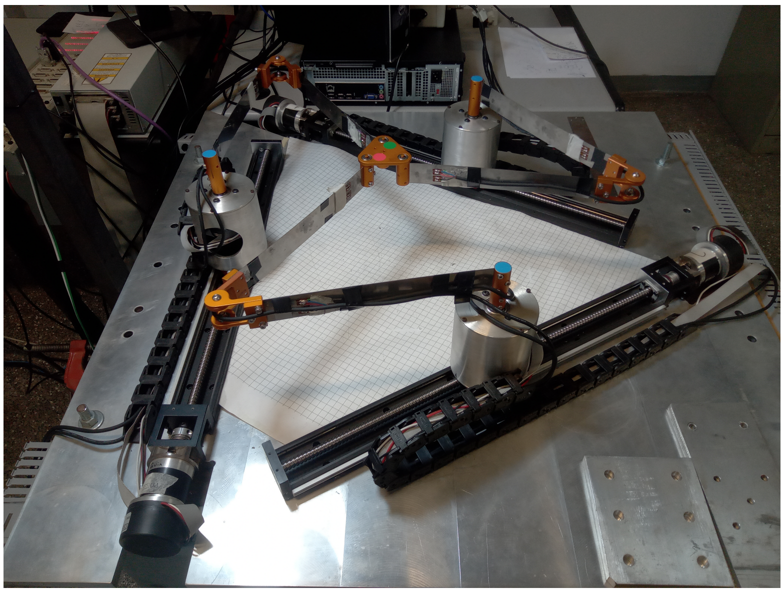

Figure 1. Current prototype

As this work will only use the 3$\underline{\text{R}}$RR mode of the prototype, the only motors ($EC60$ $flat$ da $Maxon$ brushless, with $100W$ of power and a rated current of $2.3A$, coupled to planetary gearboxes $GP52C$ with a reduction of $3.5:1$, providing a rated speed of $1200RPM$ and rated torque of $0.82Nm$) assets are $M4$, $M5$ and $M6$. These motors are responsible for providing movement to the motor joints, and for their correct functioning they have controllers (Maxon $EPOS250/5$, power supply up to $50Vdc$ and current of $5A$). Figure 2 shows a diagram of the communication. The communication between the motors and the data acquisition board ($Eletronic$ $Controller$ $Unity$, model DSPACE 1103) is done via CAN protocol, with a transmission rate of $250kbit/s$. This protocol is responsible for both motor activation and encoder data acquisition. For strain measurement, strain gauges (HBM 1-LA11K3/350-E) are used in a full bridge configuration, these are attached to each link of the manipulator to perform strain measurements for each of them. The signals collected by the strain gauge bridge need to be read by the A/D inputs of the data acquisition board, however, for this to be possible, they need to pass through a signal amplifier (HBM BM40). The current manipulator has markings, which with the help of the camera are responsible for monitoring the position of the end effector. To make this possible, the blue markings (Figure 1) made on the actuators are used to find the center of the workspace. While the red and green markings (Figure 1) are related to finding the position in relation to the center, this is a Computer Vision technique [8].

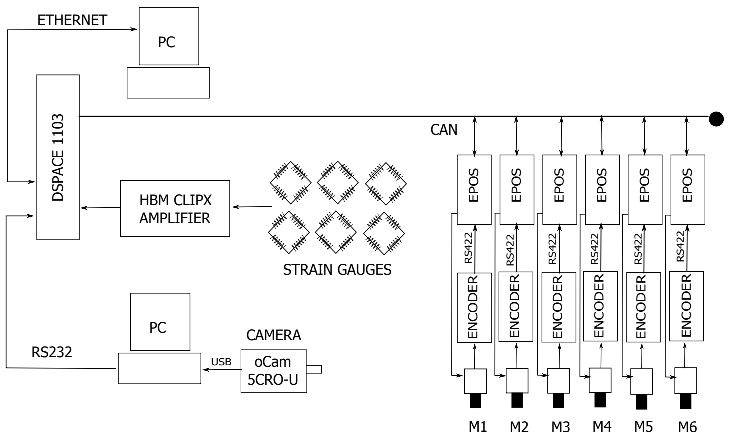

Figure 2. Instrumentation and its communication scheme

2.2 Methods

Inverse kinematics

The manipulator 3$\underline{R}$RR (Fig. 3) with rigid links can be modeled kinematics considering the geometric constraint $\left\|\overrightarrow{B_iC_i}\right\| = \|r_{C_i} - r_{B_i}\| = l_2$, where r_{C_i} and r_{B_i} are the position vectors of the passive joints C_i and B_i. This restriction imposes:

$$\left\| \begin{bmatrix} \mu_i - l_1\cos(\theta_i) \\ \rho_i - l_1\sin(\theta_i) \end{bmatrix} \right\| = \left\| l_2 \begin{bmatrix} \cos(\beta_i) \\ \sin(\beta_i) \end{bmatrix} \right\| = l_2, \tag{1}$$

where μ_i and ρ_i are defined below as

$$\begin{bmatrix} \mu_i \\ \rho_i \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - h_i \begin{bmatrix} \cos(\alpha + \eta_i) \\ \sin(\alpha + \eta_i) \end{bmatrix} - a_i \begin{bmatrix} \cos(\lambda_i) \\ \sin(\lambda_i) \end{bmatrix} - \delta_i \begin{bmatrix} \cos(\gamma_i) \\ \sin(\gamma_i) \end{bmatrix}. \tag{2}$$

So θ_i can be calculated as

$$\theta_i = 2\arctan\left(\frac{-e_{i1} \pm \sqrt{e_{i1}^2 + e_{i2}^2 - e_{i3}^2}}{e_{i3} - e_{i2}}\right), \tag{3}$$

on what $e_{i1} = -2l_{1i}\rho_i$, $e_{i2} = -2l_{1i}\mu_i$ e $e_{i3} = \mu_i^2 + \rho_i^2 + l_{1i}^2 - l_{2i}^2$.
And we also have that β_i can be given by

$$\beta_i = \arctan\left(\frac{\rho_i - l_1\sin\theta_i}{\mu_i - l_i\cos\theta_i}\right), \tag{4}$$

Artificial neural networks

Artificial neural networks (ANNs) are inspired by the structure and functioning of the nervous system, thus seeking the ability to simulate the way the human brain acquires knowledge [9]. According to Haykin [10] a neural network is a massively parallel distributed processor that has a simple processing unit and its natural function is to store the knowledge acquired through experience and make it available for use. He states that a neural network is similar to the brain in two aspects, the first being related to the fact that the network acquires knowledge of the environment around it, through a learning process. The second aspect is related to the fact that the network

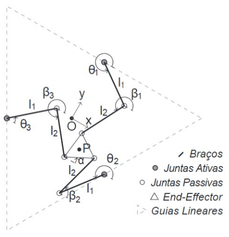

Figure 3. Schematic of 3<u>R</u>RR

has connection forces between neurons that are known as synaptic weights, used to store the acquired knowledge. When it comes to ANNs, two basic principles of building these networks should come to mind: Architecture and Learning. The architecture of an ANN concerns the type, number of processing units, and finally, the way neurons are connected, while Learning is related to the rules that are used to adjust weights in networks, and which the information is used by the rules[9]. The architecture of ANNs has three basic types: Fed-Forward Networks with Single Layer, Fed-Forward Networks with Multiple Layers and Recurrent Networks. For the present work, the use of Multi-Layer Networks was made, one of the ANNs used in the work can be seen in Fig. 4.

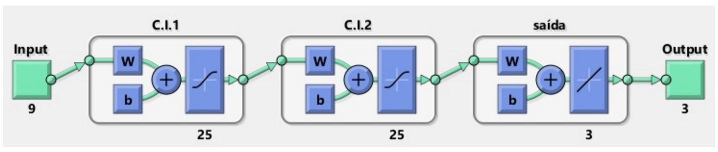

Figure 4. Multi-Layer Neural Network

In each ANN neuron there is an activation function, the most common are: linear function and sigmoid function. The sigmoid function is more suitable for problems of a nonlinear nature. Thus, for this work, we used a type of sigmoid function called hyperbilic tangent (eq. 5). for eq. 5 U is the input, while β is a constant.

$$\sigma(U) = \frac{1 - e^{-\beta U}}{1 + e^{-\beta U}}. \tag{5}$$

For ANN Learning, $Backpropagation$ was used and the Mean Square Error (MSE) was used to assess learning. Learning is supervised, since inputs and outputs are known, so an error correction learning algorithm was used. The most used learning algorithm of this type is $Backpropagation$. According to Parker [11], two types of signals are identified in the network for this type of training. One is called a Functional signal, this being an input signal (stimulus), it affects the input terminal of the network and propagates forward (neuron by neuron), until it emerges at the output terminal of the network. The other is said as Error Signal, this one originates in an output neuron of the network and propagates backwards (layer by layer). The Functional Signal traversing the network generates a response based on its current synaptic weights, this phase is known as ”forward propagation” ($forward$). The output found is compared to a desired response, and the Error Signal is propagated backwards correcting the synaptic weights, this phase is known as ”backward propagation” ($backward$). This process is repeated until the output error is within the range imposed at the beginning of the training process. The final synaptic weights obtained by the network are equivalent to the trained network, at this point it is expected that when entering a certain input, the desired response will be obtained. The EQM is used as a measure of network performance, and the lower its value, the better. The eq. 6 shows how its calculation is done.

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i^{'} - y_i)^2 \tag{6}$$

in eq. 6 $y_i^{'}$ is the desired answer, y_i is the value that seeks to be close to $y_i^{'}$ and N is the number of elements being analyzed.

Description of the problem situation and work methodology

For the problem in question, we want to find a function F that describes the model, presented in eq. 7.

$$(x, y, \alpha) = F(\theta_1, \theta_2, \theta_3, s1, s2, s3, s4, s5, s6). \tag{7}$$

In this model x, y and α refer to the position of the end effector. θ_1, θ_2 and θ_3 deal with the angular displacement of the active joints, while s_1, s_2, s_3, s_4, s_5 and s_6 deal with the deformation of each of the manipulator links. The F function is a non-linear function, finding this function analytically would be a complex job. In Machado's work [12] this can be verified for the modeling of a single link. The present work seeks to find an Artificial Neural Network that reproduces the function F. The work follows a work methodology that is divided into three main parts: Obtaining Data, Data Treatment and Training Neural Networks. All simulations performed in this work, in each of these three stages, made use of the MATLAB software.

3 Results

3.1 Obtaining Experimental Data

To obtain the experimental data, the instrumentation and the communication model presented in Figure 2 were used. Torques of -10°, 10°, -20° and 20° were applied to the manipulator motors 3$\underline{\text{R}}$RR (Figure 1), these torques could be applied to one motor individually, in two motors simultaneously, or on all three engines simultaneously. An example of system activation could be -10° $M4$, 20° $M5$ and 0° $M6$, activations of this type were made 32 times at random and with the initial position of the end effector also random. Thus, for each triggering, a data matrix of dimension 12 X 10001 points was obtained, with the twelve rows of the matrix corresponding to the values of $x, y, \alpha, e_1, e_2, e_3, s_1, s_2, s_3, s_4, s_5, s_6$, where in each column a value was recorded for each variable every 1 ms, during 10.001 s of operation, totaling 10001 data points.

3.2 Treatment of Experimental Data

The data collected by the system needed to be treated before being inserted into the Neural Networks. One of the problems presented in the data collection was an excess of irrelevant data for the training of the networks, these data comprised the moment before the manipulator operation and the moment after its operation, removing these data the width of the matrices dropped from 10001 points to 5176, a reduction of nearly half the data. The rest of the data processing was done according to the type of data used.

For the data on the position of the end-effector and strain gauges, digital low-pass filters were used, an idea used by Félix [13] in his work. The use of this filter was responsible for a smoothing of the data.

The encoders used are incremental, so they do not have an initial reference. With that, the inverse kinematics of the rigid manipulator was used, in order to calculate the initial reference with the aid of Computer Vision [8], later these data were added to the incremental data of the encoder to find the angular displacement. At the end of the treatment, it was verified that of the 32 operations performed, only 28 were valid.

3.3 Neural Network Training

For the work in question, so far, the use of Multi-Layer Fed Forward Networks has been used. A first strategy used was to group all trajectories into a single one, and use it for training the neural network, dividing the points of this trajectory into three groups: training, validation and testing, aiming to use the cross-validation method. In order to analyze the performance of a Neural Network in relation to its own variables: training set size, number of layers and number of neurons per layer. For this, a scheme was made, where training sets with 1 trajectory, 8 trajectories, 15 trajectories and 22 trajectories were used, 1 or 2 hidden layers and each of these layers ranging from 1 to 10 neurons. All possible combinations between these network variables were performed, the result is shown in Fig. 5.

The graph shows that the more trajectories are added to the network, the worse its performance becomes (mean squared error increases). For training with 1 trajectory, 8 trajectories, 15 trajectories and 22 trajectories the best results for the test data were respectively: $1.02 * 10^{-6}$, $5.59 * 10^{-4}$, $2.02 * 10^{-3}$ and $9.7 * 10^{-3}$. Fig.5 shows that the performance values also improve as the network complexity increases (more layers and more neurons

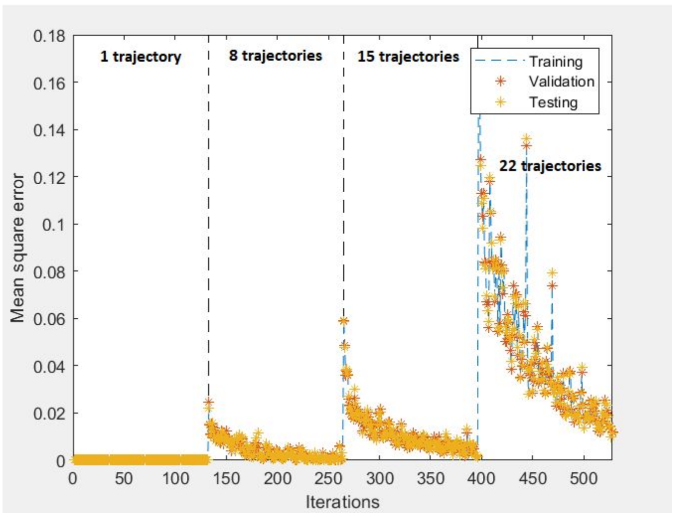

Figure 5. Variation of neural network parameters

per layer). For a fixed number of trajectories, there is a decrease in the value of the MSE, that is, an increase in performance.

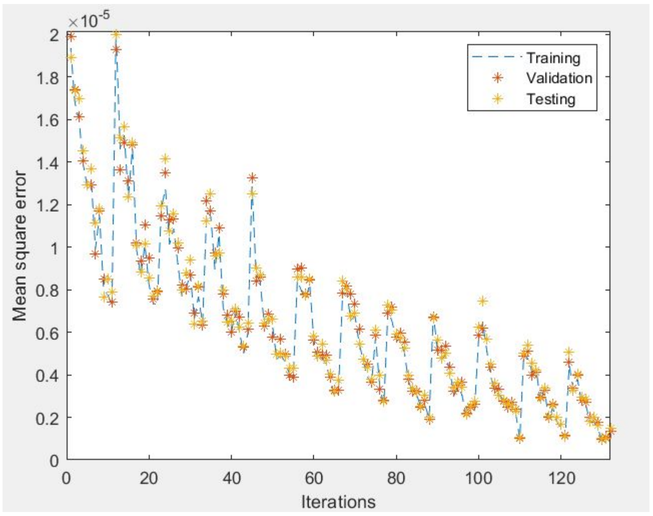

Figure 6. Variation of neural network parameters for a trajectory

In Fig. 6 it shows the decrease of the MSE value for the training of a trajectory. It is possible to observe that for a single trajectory the performance results were the best, so it was decided to train the 28 available neural networks and verify if, by training all of them, it would be possible to reach a network that generalizes to all the others. For this, the neural network illustrated in Fig. 4, the result for the first six trajectories is shown in Tab. 1.

Although the data presented only illustrate the first six trajectories, they are not very different from the rest, and it does not make sense to include them all here. For the same trajectory that is being trained, the error is low, in the order of 10^{-8} to 10^{-6}, while for the others sometimes the MSE is a value greater than unity. With this, it is possible to conclude that this method was not enough to obtain a generalist network.

Table 1. MSE (Trajectories x Networks)

Trajectory	net_1	net_2	net_3	net_4	net_5	net_6
1st	3.3248e-08	0.1754	32.9263	0.9201	0.6137	0.2810
2nd	0.0520	4.3655e-08	16.1981	0.2794	0.3825	0.7424
3rd	0.1819	0.1915	4.1479e-06	0.8568	0.6999	0.3051
4th	0.1986	0.1708	44.9290	9.5972e-07	1.1229	0.2951
5th	0.5008	0.4001	63.4788	0.4774	9.4237e-08	0.0579
6th	0.4843	0.3371	38.4117	0.6701	0.1251	4.2879e-08

4 Conclusions

So far, the use of Multi-Layer Neural Network is not proving to be adequate, it is not yet known if this is happening due to a characteristic of the network itself, or if the problem is in the way the data is being treated. For future work, a next network architecture that will be used is the Recurrent Networks, it is expected to achieve a network that can make the generalization.

References

[1] J.-P. Merlet. *Parallel robots*, volume 128. Springer Science & Business Media, 2005.
[2] Y. Li and G. M. Bone. Are parallel manipulators more energy efficient? In *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No. 01EX515)*, pp. 41–46. IEEE, 2001.
[3] F. Paccot, N. Andreff, and P. Martinet. A review on the dynamic control of parallel kinematic machines: Theory and experiments. *The International Journal of Robotics Research*, vol. 28, n. 3, pp. 395–416, 2009.
[4] H. S. Bidokhti and J. Enferadi. Direct kinematics solution of 3-rrr robot by using two different artificial neural networks. In *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, pp. 606–611. IEEE, 2015.
[5] A. Elsheikh, E. Showaib, and A. Asar. Artificial neural network based forward kinematics solution for planar parallel manipulators passing through singular configuration. *Adv Robot Autom*, vol. 2, n. 106, pp. 2, 2013.
[6] A. Moori, J. Khoramdel, and S. A. A. Moosavian. Deep learning approach for object tracking of roboeye. In *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*, pp. 386–391. IEEE, 2019.
[7] A. Csiszar, J. Eilers, and A. Verl. On solving the inverse kinematics problem using neural networks. In *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pp. 1–6. IEEE, 2017.
[8] F. T. Colombo, de J. V. Carvalho Fontes, and da M. M. Silva. A visual servoing strategy under limited frame rates for planar parallel kinematic machines. *Journal of Intelligent & Robotic Systems*, vol. 96, n. 1, pp. 95–107, 2019.
[9] K. Faceli. *Inteligência Artificial Uma Abordagem de Aprendizado de Máquina*, volume 1. genio, 2011.
[10] S. Haykin. *Redes Neurais Princípios e Prática*, volume 2. genio, 2001.
[11] D. Parker. Optimal algorithms for adaptive networks: Second order back propagation, second order direct propagation, and second order hebbian learning. *IEEE 1st International Conference on Neural Networks*, vol. 2, pp. 593–600, 1987.
[12] C. C. Machado. Modelagem matemática e controle ativo de um manipulador com um elo flexível, 2007.
[13] L. Félix. Instrumentation of a parallel manipulator with flexible links: a neural network application. *COBEM 2021, 26^a international Congress of Mechanical Engineering*, pp. 1–9, 2021.