Chapter 5
Optimization of Training Data Set Based on Linear Systematic Sampling to Solve the Inverse Kinematics of 6 DOF Robotic Arm with Artificial Neural Networks

Ma. del Rosario Martínez-Blanco, Teodoro Ibarra-Pérez, Fernando Olivera-Domingo, and José Manuel Ortiz-Rodríguez

5.1 Introduction

The large amount of data available today constitutes one of the most valuable capital for organizations, because through its analysis, it is possible to obtain strategic information for decision-making, detection of behaviors, establishment of predictive models, among others [1].

The current volume of information has exceeded the processing capabilities of current conventional systems. The intervention of new processing algorithms and scalable techniques, which allow fast and efficient information processing, is increasingly necessary [2].

When having a very large amount of data, there are usually two ways to approach the problem. One way can be to redesign the algorithms without affecting performance to obtain an efficient execution with all the data. A second approach may involve reducing the data set to obtain a very similar result as if the entire volume of data were used [3].

Success in knowledge extraction algorithms is highly dependent on the integrity and consistency of the extracted data. Particularly in the field of artificial neural networks, most research focuses its efforts on specific applications and training algorithms that improve precision and convergence of results. However, most of the studies do not describe the procedure that was applied in the data preprocessing

M. d. R. Martínez-Blanco · T. Ibarra-Pérez · J. M. Ortiz-Rodríguez (✉)
Laboratorio de Innovación y Desarrollo Tecnológico en Inteligencia Artificial (LIDTIA), Universidad Autónoma de Zacatecas (UAZ), Zacatecas, México

T. Ibarra-Pérez · F. Olivera-Domingo
Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas (UPIIZ), Instituto Politécnico Nacional (IPN), Zacatecas, México

S. Paul et al. (eds.), *Frontiers of Data and Knowledge Management for Convergence of ICT, Healthcare, and Telecommunication Services*, EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-77558-2_5

stage and, in general, the determination of success or failure in the knowledge extraction algorithms is influenced by the quality of the training data [4].

Data preprocessing is a stage that can increase the quality and reliability of the data because a low quality in the data leads to a low quality in the knowledge extracted. Although it is true that a very complete training data set would allow a better understanding of the problem to be obtained, the training time required will be much longer and computationally costly, making it infeasible [5].

Sampling is one of the most appropriate preprocessing methods to solve this problem due to its advantages in performance and low processing cost required during its application in knowledge extraction algorithms in various fields of engineering, statistics, machine learning, and data mining [6].

Data preprocessing techniques focus on two areas: data preparation and data reduction. Data preparation is mandatory and refers to the adequacy of the data so that the algorithms can be executed correctly, such as normalization, cleaning, and probably the recovery of lost data. On the other hand, data reduction is not always required and refers to the generation of a reduced size that maintains the integrity of the information as much as possible, such as the selection of characteristics, the selection of instances, the grouping, and sampling among others [7, 8].

Linear Systematic Sampling (LSS) is one of the most used techniques thanks to its ease of use. It was first introduced in 1944 by [9] and is also known as Cluster Sampling method, which consists of dividing the population N into k groups of n elements each, allowing all the elements of the data set to have the same chances of being selected [10, 11].

In this study, a Reduction Data Filter (RDF) algorithm based on the LSS method was implemented to process more than 4 billion data in a conventional AMD Ryzen 7 processor with 16 GB of RAM. In this way, a training data set was generated, which was used by two different Artificial Neural Networks (ANN) architectures to analyze the performance and generalizability of both architectures.

5.1.1 Artificial Neural Networks

The understanding of our brain has allowed the creation of Artificial Neural Networks (ANN) due to the fact that they are inspired by its operation, while a microprocessor processes information sequentially, the human brain does it in parallel and concurrently, however, we are still far from emulating the simplest capabilities of human reasoning, although ANN is currently a very powerful instrument for various applications in engineering and a very promising field of research [12].

An artificial neuron can be described as a computational structure or a mathematical abstraction model inspired by the neurons of our brain to which input signals arrive, as occurs with the dendrites of a real neuron, and generates an output signal, as happens with the axon [13]. An example of the schematic of an artificial neuron is shown below in Fig. 5.1.

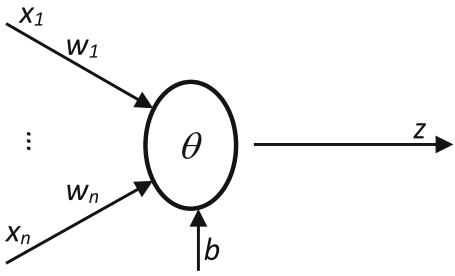


Fig. 5.1 Scheme of an artificial neuron

Each of the inputs $x_{1,\ldots,}\ x_n$ is assigned a value or synaptic weight $w_{1,\ldots,}w_n$. The output of the perceptron is an independent function of the value of its inputs, with their respective weights and threshold value. The weights w_i represent the intensity of synapses connecting two neurons and can be negative or positive. θ is the transfer function or firing threshold from which the neuron is activated [14].

The output is calculated by the cumulative sum of the product of all the input signals multiplied by their corresponding synaptic weights plus a bias, b, as shown below in Eq. (5.1).

$$z = \sum_i x_i \times w_i + b \tag{5.1}$$

The output generated is used as input to a transfer function where the response to the artificial neuron is generated, as shown below in Eq. (5.2).

$$y_i = f\left(\sum_i x_i \times w_i + b\right). \tag{5.2}$$

ANNs are interconnected neurons distributed in parallel. Generally, the form of connection between neurons determines the type of network and they are usually grouped in layers. A layer is a set of neurons and according to its location it can be the input layer, a hidden layer, or the output layer. The architecture of a neural network is determined according to the arrangement of neurons within the layers and the forms of connection between them [14, 15].

Figure 5.2 shows a simplified model that describes an ANN with three layers, where it is observed that the network has R^1 inputs, i^1 neurons in the first layer, i^2 neurons in the second layer, and i^3 neurons in the third layer. The bias constant with a value of 1 is added in each neuron and in this way a layered approach can be obtained to analyze the complete structure of the network [16].

According to the feedback existing in the network, it can be seen that, if there is no connection between the output layer and the neurons of the input layer, the network does not maintain a previous memory state, so it is a forward propagation network. On the other hand, when there is feedback between the input and output

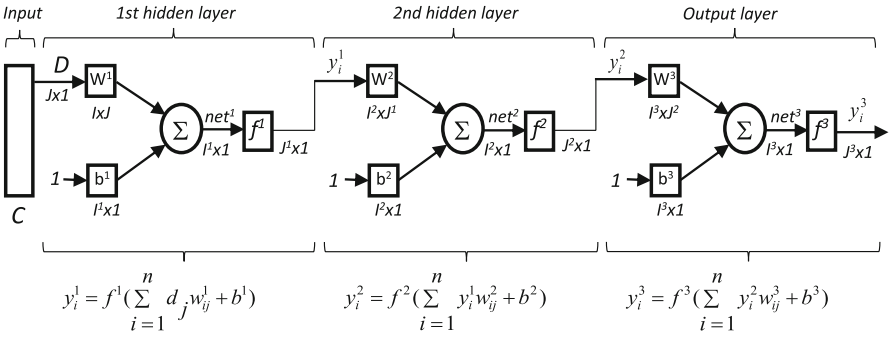


Fig. 5.2 Artificial Neural Network with two hidden layers

layers, a memory of their previous states is kept and the next state depends not only on the input signals but also on the previous states of the network, generating a backward propagation network [14].

In 1986 Rumelhart and McCelland proposed the Back Propagation Neural Network (BPNN) whose fundamental architecture consists of three layers. In this type of network, the number of layers that are necessary can be used and its mathematical foundation is based on the gradient descent algorithm, where the synaptic weights constantly change when processed by neurons through an activation function, producing outputs for the next layer until the minimum error is reached [14, 17, 18].

There is generally a problem with this type of network because the structural parameters must be established at the beginning of the training. Currently there is no procedure that guarantees the optimal configuration of these parameters and they are usually proposed according to the researcher's previous experience in trial and error experiments [19].

In Generalized Regression Neural Network (GRNN) architectures, unlike BPNN, it is not necessary to optimize the training parameters such as the learning rate and the momentum, on the contrary, the smoothing factor or propagation value of the network is determined. This value must be greater than zero and can generally be in the range of 0.01 to 1, obtaining excellent results. To determine the propagation value, it is necessary to estimate the probability density function according to the samples used in the training of the network and to carry out several experiments in order to determine the most appropriate value to train the GRNN [20, 21].

Donald F. Spech introduced this type of network in 1991, which is capable of producing continuous output values and rapidly draining sparse data sets. Due to the fact that only one forward propagation is necessary, the training of this type of network is very fast compared to the BPNN, where the information must be backward and forward propagated several times until an acceptable value of the desired error is found [20, 22].

5.1.2 *Inverse Kinematics Solution with Artificial Neural Networks*

Inverse kinematics is one of the most interesting problems in the field of industrial robotics. Inverse kinematics consists of determining the joint values of the end effector for a certain position and orientation in a Cartesian plane. This problem can be solved by means of a closed solution, since on some occasions the solution in real time is necessary for applications such as tracking a trajectory. However, the solution could be very complex due to the geometry of the manipulator, forcing the implementation of iterative solutions, which would be unfeasible for real-time applications, requiring the intervention of appropriate predictive models in the field of soft computing, where ANNs are one of those techniques that can be used to obtain acceptable results [23–25].

During the last decades, researchers have shown a special interest in the use of ANN, particularly due to its characteristics related to nonlinearity, parallelism, high robustness, fault tolerance, and great capacity for learning and generalization from complex and nonlinear examples [26–29].

The multilayer perceptron trained with the back propagation algorithm (BP) is one of the most widely used network architectures in modeling, optimization, and classification applications [19, 30–32]. This type of network is one of the most frequently used to learn the equations of the direct and inverse kinematics of 6 Degrees of Freedom (DOF) robotic manipulators, where the network learns the functional relationship between the input space and the output space through supervised training, based on a mapping adapting the solution in a nonlinear relationship between the locations of the end effector with the desired location [23, 33].

Currently, the determination of the structural parameters in the use of ANN continues to be a difficult and complicated task, because the parameters are generally defined by the researcher's previous experience in trial and error procedures, investing large amounts of time and resources, without guaranteeing the optimal configuration of the structural parameters [19, 28, 34, 35].

In this study, the Robust Design Artificial Neural Networks (RDANN) methodology was used to determine the optimal parameters in the first proposed network architecture: BPNN. Likewise, a systematic procedure was used to calculate the optimum value in the second proposed network architecture: GRNN. Both networks were used to solve the Inverse Kinematics (IK) of a 6 DOF robotic manipulator after training them with two data sets of the same size, but with different preprocessing characteristics, with the aim of analyzing the performance and generalizability of the proposed networks based on the quality of the training data [19, 36].

5.2 Neural Networks Based Inverse Kinematics Solution

5.2.1 Kinematics Analysis of Ketzal Robot

The morphology of robotic manipulators generally refers to the shape of the components and their structural mechanical parts [37]. A robotic manipulator generally has rigid mechanisms known as links that are connected by prismatic or rotating joints forming an open chain that can be operated by actuators [38].

In this study, the structure of a robotic manipulator called Ketzal with six DOF was used. Figure 5.3 shows the structure and coordinates reference systems of the robot, which is based on an open source, low-cost, and 3D printed project [39].

The forward kinematics calculation is about finding the position and the orientation vectors of the end effector with respect to a fixed coordinate reference system, given the vector of joint angles [40]. The inverse kinematics problem is about calculating the vector of joint angles given the position and orientation vectors of the end effector with respect to a fixed coordinate reference system [41].

The forward kinematics calculation results in an homogeneous transformation matrix T of size 4x4, as shown in Eq. (5.3), where the spatial configuration between the joints of the manipulator is related to the position and orientation with respect to a fixed reference system [24].

$$T_0^6 = \begin{bmatrix} R_0^6 & P_0^6 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

where R_0^6 represents a 3×3 size rotation matrix composed of the vectors n, o y a and that defines the orientation of the end effector and P_0^6 is the position vector of the end effector in the coordinate reference system. In this chapter, the Denavit-Hartenberg (D-H) method was used to calculate the forward kinematics of the Ketzal manipulator by implementing our basic transformations [42]. The D-H parameters are shown below in Table 5.1.

Fig. 5.3 Ketzal robotic manipulator

Table 5.1 Ketzal robot D-H parameters

Link offset (cm)	Joint angle (rad)	Link length (cm)	Twist angle (rad)
$d_1 = 20.2$	$\theta_1 = q_1$	$a_1 = 0$	$\alpha_1 = \frac{\pi}{2}$
$d_2 = 0$	$\theta_2 = q_2$	$a_2 = 16$	$\alpha_2 = 0$
$d_3 = 0$	$\theta_3 = q_3 + \frac{\pi}{2}$	$a_3 = 0$	$\alpha_3 = \frac{\pi}{2}$
$d_4 = 19.5$	$\theta_4 = q_4$		

The transformations, which depend on the configurations of the links, consist of a succession of rotations and translations allowing to relate the reference system of element *i* with the system of element *i-1*, which is given by Eq. (5.4) [43].

$$^{i-1}A_i = \text{Rot}_{z\theta i}\text{Trans}_{x,di}\text{Trans}_{x,\alpha i}\text{Rot}_{z,\alpha i} \tag{5.4}$$

Similarly,

$$^{i-1}A_i = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

where i is the number of the link, θ_i is the angle of rotation of the joint, α_i is the rotation of the joint, a_i is the length of the link, d_i is the displacement of the link, and $C\theta_i = \cos(\theta_i)\, y\, S\theta_i = \sin(\theta_i)$.

Realizing the product of the six matrices obtained from Eq. (5.6), the matrix that indicates the location of the final system with respect to a reference system located at the base of the robot is obtained, which is known as the homogeneous transformation matrix T_0^6 [43].

$$T_6^0 = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3 \cdot {}^3A_4 \cdot {}^4A_5 \cdot {}^5A_6 \tag{5.6}$$

where

$$^0A_1 = \begin{bmatrix} \cos(q_1) & 0 & \sin(q_1) & 0 \\ \sin(q_1) & 0 & -\cos(q_1) & 0 \\ 0 & 1 & 0 & a_0 + a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.7}$$

$$^1A_2 = \begin{bmatrix} \cos(q_2) & -\sin(q_2) & 0 & a_2 * \cos(q_2) \\ \sin(q_2) & \cos(q_2) & 0 & a_2 * \sin(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.8}$$

$$
{}^{2}A_3 = \begin{bmatrix} -\sin(q_3) & 0 & \cos(q_3) & 0 \\ \cos(q_3) & 0 & \sin(q_3) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.9}
$$

$$
{}^{3}A_4 = \begin{bmatrix} \cos(q_4) & 0 & -\sin(q_4) & 0 \\ \sin(q_4) & 0 & \cos(q_4) & 0 \\ 0 & -1 & 0 & a_3 + a_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.10}
$$

$$
{}^{4}A_5 = \begin{bmatrix} \cos(q_5) & 0 & \sin(q_5) & 0 \\ \sin(q_5) & 0 & -\cos(q_5) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.11}
$$

$$
{}^{5}A_6 = \begin{bmatrix} \cos(q_6) & -\sin(q_6) & 0 & 0 \\ \sin(q_6) & \cos(q_6) & 0 & 0 \\ 0 & 0 & 1 & a_5 + a_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.12}
$$

Getting,

$$
\begin{aligned} n_x = {} & \sin(q_1)\sin(q_6)\cos(q_4) + \sin(q_1)\sin(q_4)\cos(q_5)\cos(q_6) + \sin(q_2+q_3) \\ & \sin(q_4)\sin(q_6)\cos(q_1) - \sin(q_5)\cos(q_1)\cos(q_2+q_3)\cos(q_6) - \sin(q_2+q_3) \\ & \cos(q_1)\cos(q_4)\cos(q_5)\cos(q_6) \end{aligned} \tag{5.13}
$$

$$
\begin{aligned} n_y = {} & -\sin(q_6)\cos(q_1)\cos(q_4) - \sin(q_4)\cos(q_1)\cos(q_5)\cos(q_6) + \sin(q_1) \\ & \sin(q_2+q_3)\sin(q_4)\sin(q_6) - \sin(q_1)\sin(q_5)\cos(q_2+q_3)\cos(q_6) - \sin(q_1) \\ & \sin(q_2+q_3)\cos(q_4)\cos(q_5)\cos(q_6) \end{aligned} \tag{5.14}
$$

$$
\begin{aligned} n_z = {} & -\sin(q_2+q_3)\sin(q_5)\cos(q_6) + \cos(q_2+q_3)\cos(q_4)\cos(q_5)\cos(q_6) \\ & -\sin(q_4)\sin(q_6)\cos(q_2+q_3) \end{aligned} \tag{5.15}
$$

$$o_x = \sin(q_1)\cos(q_4)\cos(q_6) + \sin(q_2+q_3)\sin(q_4+q_6)\cos(q_1) - \sin(q_1)\sin(q_4)\sin(q_6)\cos(q_5) - \sin(q_5)\sin(q_6)\cos(q_1)\cos(q_2+q_3) \tag{5.16}$$

$$o_y = \sin(q_4)\sin(q_6)\cos(q_1)\cos(q_5) + \sin(q_1)\sin(q_2+q_3)\sin(q_4)\cos(q_6) + \sin(q_1)\sin(q_5)\sin(q_6)\cos(q_2+q_3) - \sin(q_1)\sin(q_2+q_3)\sin(q_6)\cos(q_4)\cos(q_5) - \cos(q_1)\cos(q_4)\cos(q_6) \tag{5.17}$$

$$o_z = \sin(q_2+q_3)\sin(q_5)\sin(q_6) - \sin(q_6)\cos(q_2+q_3)\cos(q_4)\cos(q_5) - \sin(q_4)\cos(q_2+q_3)\cos(q_6) \tag{5.18}$$

$$a_x = \sin(q_1)\sin(q_4)\sin(q_5) - \sin(q_2+q_3)\sin(q_5)\cos(q_1)\cos(q_4) + \cos(q_1)\cos(q_2+q_3)\cos(q_5) \tag{5.19}$$

$$a_y = -\sin(q_4)\sin(q_5)\cos(q_1) - \sin(q_1)\sin(q_2+q_3)\sin(q_5)\cos(q_4) + \sin(q_1)\cos(q_2+q_3)\cos(q_5) \tag{5.20}$$

$$a_z = \sin(q_2+q_3)\cos(q_5) + \sin(q_5)\cos(q_2+q_3)\cos(q_4) \tag{5.21}$$

$$p_x = \left[\sin(q_1)\sin(q_4)\sin(q_5) - \sin(q_2+q_3)\sin(q_5)\cos(q_1)\cos(q_4) + \cos(q_1)\cos(q_2+q_3)\cos(q_5)\right](a_5+a_6) + \cos(q_1)\cos(q_2+q_3)(a_3+a_4) + \cos(q_1)\cos(q_2)a_2 \tag{5.22}$$

$$p_y = -\Big[\sin(q_4)\sin(q_5)\cos(q_1) - \sin(q_1)\cos(q_2+q_3)\cos(q_5) + \sin(q_1)\sin(q_2+q_3)\sin(q_5)\cos(q_4)\Big](a_5+a_6) + \sin(q_1)\cos(q_2+q_3)(a_3+a_4) + \sin(q_1)\cos(q_2)a_2 \tag{5.23}$$

$$p_z = [\sin(q_2+q_3)\cos(q_5) + \sin(q_5)\cos(q_2+q_3)\cos(q_4)](a_5+a_6) + \sin(q_2+q_3)(a_3+a_4) + \sin(q_2)a_2 + (a_0+a_1) \tag{5.24}$$

It can be seen that Eqs. (5.13, 5.14, 5.15, 5.16, 5.17, 5.18, 5.19, 5.20, and 5.21) reflect the values of the orientation vectors of the end of the robot [*noa*] as a function of the joint coordinates ($q_1, q_2, q_3, q_4, q_5, q_6$) and Eqs. (5.22), (5.23) y (5.24) reflect the values of the position vector of the end of the manipulator (p_x, p_y, p_z) as a function of the joint coordinates and lengths of the links ($a_0, a_1, a_2, a_3, a_4, a_5, a_6$).

5.2.2 *Description of Data Sets*

According to the geometry and physical dimensions of the Ketzal robotic manipulator, the workspace can be represented by the set of position coordinates, orientation, and joint values. In principle, the workspace is made up of an infinite set of spatial coordinates. However, it is necessary to generate a space defined by a finite set so that it can be processed by a computer, because the volume in the data considerably influences the available processing capacities [4].

Two data sets *A* and *B* were proposed. Both sets were generated from the transformation matrices described by Eq. (5.5). According to the geometric characteristics of the Ketzal robot, the ranges of movement for each of the joints ($\Theta_1 \ldots \Theta_6$) are described in Table 5.2.

The workspace for both proposed sets is the same; however, the distribution of the spatial coordinates is different due to the resolution used in the ranges of motion initially established in each of the joints during the generation of the two data sets. In other words, the amount of data generated is defined as a function of the spatial resolution established in the range of joint values. For example, joint Θ_1 has a range of motion from 0 to 2π; therefore, if a jump is set $\Delta\theta_1 = \pi$, only three values [0, π, 2π] are considered, on the contrary, if a jump is used from $\Delta\theta_1 = \pi/5$ a better spatial resolution is obtained considering eleven values [15].

Table 5.2 Angular ranges in the Ketzal robot joints

(rad)	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Mínimum	0	0	2π	0	2π	0
Máximum	2π	π	$\frac{\pi}{2}$	2π	$\frac{\pi}{2}$	2π

Available hardware features and capabilities are one of the most important factors that significantly influence data processing. Below a description of the two data sets that were generated by implementing two arrays of *rows* × *columns* is presented [44].

$$\text{Dataset}A = \begin{bmatrix} a_{11}\ a_{12}\ \ldots\ a_{1c} \\ a_{21}\ a_{22}\ \ldots\ a_{2c} \\ \ldots\ \ldots\ \ldots\ \ldots \\ a_{r1}\ a_{r2}\ \ldots\ a_{rc} \end{bmatrix} \tag{5.25}$$

For data set A, the subscript r represents the number of data generated by each variable with a value of 24,000 double precision data, with a spatial resolution of 10×5×5×6×4×4, that is, joint Θ_1 within its range of motion can only generate 10 values, the second joint 5 values, and so on for the other joints. The subscript c represents the total number of variables used with a value of 18 variables, giving a total of 432,000 data with a physical space in memory of 3.29 MB. The elements $\{a_{r1}, a_{r2}, a_{r3}\}$ correspond to the position vector $[p] = \{p_x, p_y, p_z\}$, the elements $\{a_{r4} \ldots a_{r12}\}$ correspond to the orientation vector $[noa] = \{n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z\}$, and finally the elements $\{a_{r13} \ldots a_{r18}\}$ correspond to the vector of joint values $[\Theta] = \{\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \Theta_6\}$.

$$\text{Dataset}B = \begin{bmatrix} b_{11}\ b_{12}\ \ldots\ b_{1c} \\ b_{21}\ b_{22}\ \ldots\ b_{2c} \\ \ldots\ \ldots\ \ldots\ \ldots \\ b_{r1}\ b_{r2}\ \ldots\ b_{rc} \end{bmatrix} \tag{5.26}$$

For data set B, the subscript r has a value of 244,140,625 double precision data, with a spatial resolution of 25×25×25×25×25×25, where each of the joints can generate 10 values within its range of motion. The subscript c has a value of 18 variables giving a total of 4,394,531,250 data with a physical space in memory of 4.09 GB.

The elements $\{b_{r1}, b_{r2}, b_{r3}\}$ correspond to the position vector $[p] = \{p_x, p_y, p_z\}$, the elements $\{b_{r4} \ldots b_{r12}\}$ correspond to the orientation vector $[n\ o\ a] = \{n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z\}$, and finally the elements $\{b_{r13} \ldots b_{r18}\}$ correspond to the vector of joint values $[\Theta] = \{\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \Theta_6\}$.

5.2.3 Data Set Collection

To collect the data sets from the kinematic analysis of the proposed manipulator, a six-dimensional matrix was generated that contains all the combinations of values defined in Θ_1, Θ_2, Θ_3, Θ_4, Θ_5, and Θ_6. For data set B, each of the joints was defined with 25 values; therefore, the resulting matrix has a dimension of

25x25x25x25x25x25. For data set A, the resulting matrix has a dimension of 10x5x5x6x4x4. The x, y, z coordinates are deduced through the direct kinematics equations, starting from the ranges of motion for each data set as described in Table 5.2 and taking into account the lengths of the six links of the manipulator, where $a_0 = 10.1$ cm, $a_1 = 10.1$ cm, $a_2 = 16.0$ cm, $a_3 = 9.2$ cm, $a_4 = 10.3$ cm, $a_5 = 1.345$ cm, and $a_6 = 5.37$ cm. When calculating the direct kinematics Eqs. (5.13, 5.14, 5.15, 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, 5.22, 5.23, and 5.24) as a function of the joints and lengths of the links, a six-dimensional matrix is obtained for each equation solved. Finally, to obtain the data set, the result of the direct kinematics equations is grouped together with the joint value matrix in a single matrix. The resulting matrix contains nine orientation matrices, three position matrices, and six joint value matrices, for a total of 18 six-dimensional matrices.

5.2.4 *Dispersion Analysis of the Generated Data Set*

Starting from the initial position and according to the range of movement defined for each of the joints, the workspace can be appreciated from different perspectives by using scatter diagrams in the position vectors as a function of the joints. Figure 5.4 shows the dispersion of the position data as a function of the three joints of the end effector, generating a half sphere as shown below.

By including the combinations of the q_1 joint with the previous one, a set of half spheres following a circular trajectory is obtained, because the joint has a range of motion from 0 to 360 degrees as seen below in Fig. 5.5.

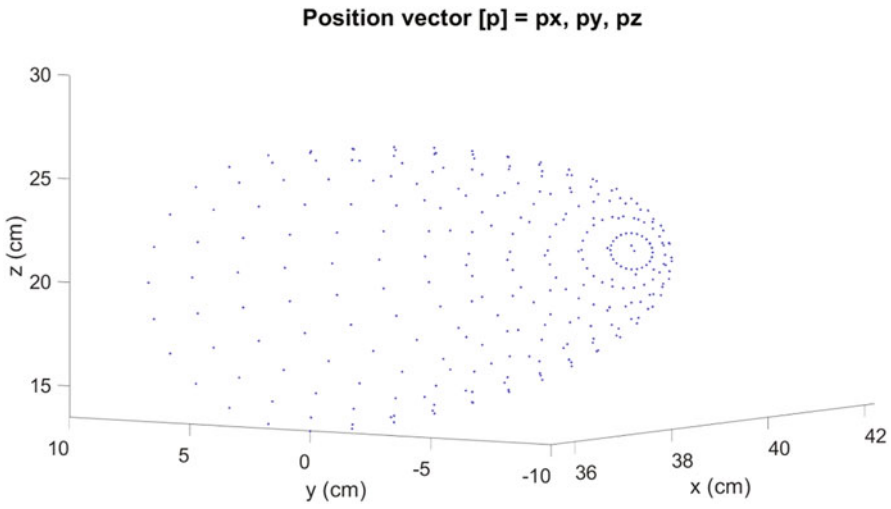


Fig. 5.4 Position vector (p_x, p_y, p_z) in function of joints q_4, q_5 **and** q_6

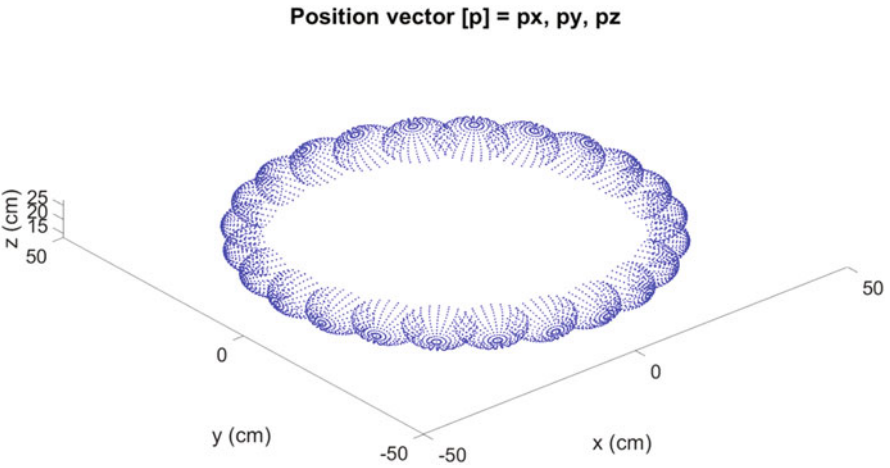


Fig. 5.5 Position vector $(\boldsymbol{p_x}, \boldsymbol{p_y}, \boldsymbol{p_z})$ in function of joints $\boldsymbol{q}_1$, $\boldsymbol{q}_4$, $\boldsymbol{q}_5$ **and** $\boldsymbol{q}_6$

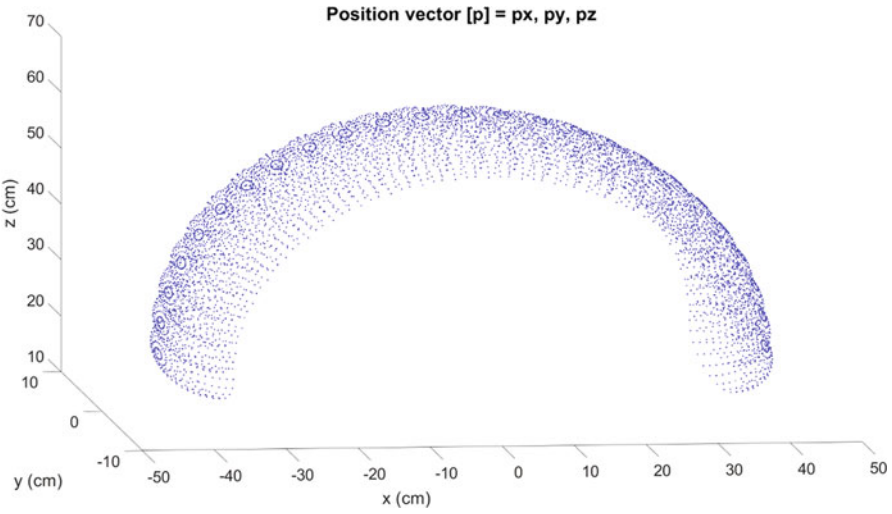


Fig. 5.6 Position vector $(\boldsymbol{p_x}, \boldsymbol{p_y}, \boldsymbol{p_z})$ in function of joints $\boldsymbol{q}_2$, $\boldsymbol{q}_4$, $\boldsymbol{q}_5$ **and** $\boldsymbol{q}_6$

On the contrary, if the combinations of the q_2 joint instead of the q_1 joint are included in the position vector functions(p_x, p_y, p_z), a set of half spheres in a vertical plane following a half-circle trajectory is obtained, because the joint q_2 has a range movement from 0 to 180 degrees as shown below in Fig. 5.6.

If the combinations of q_3, q_4, q_5 and q_6 joints are included in the position vector functions (p_x, p_y, p_z), a series of half spheres in a horizontal plane following a circular path are obtained, because the joint q_3 has a range of motion of -90 to 90 degrees as shown below in Fig. 5.7.

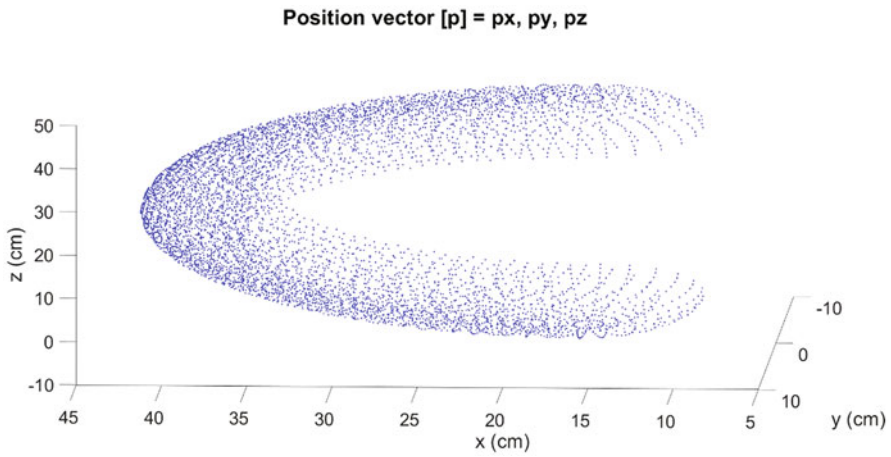


Fig. 5.7 Position vector $(\boldsymbol{p_x}, \boldsymbol{p_y}, \boldsymbol{p_z})$ in function of joints $\boldsymbol{q}_3$, $\boldsymbol{q}_4$, $\boldsymbol{q}_5$ **and** $\boldsymbol{q}_6$

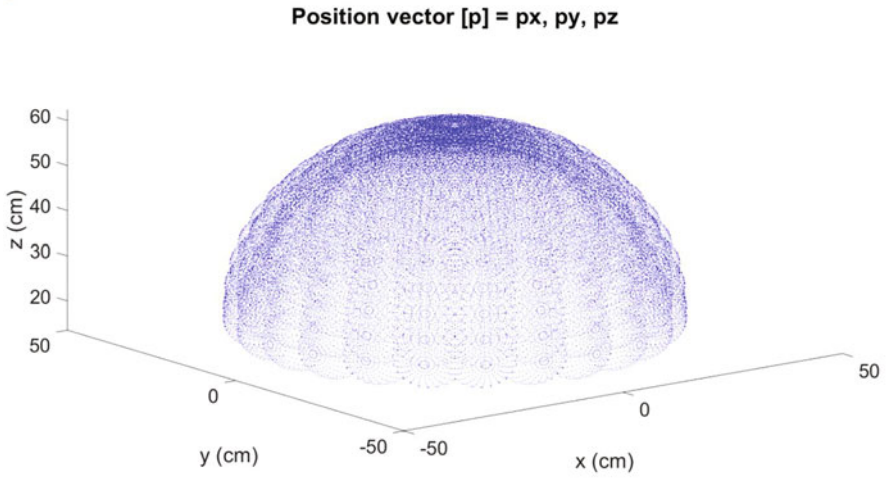


Fig. 5.8 Position vector $(\boldsymbol{p_x}, \boldsymbol{p_y}, \boldsymbol{p_z})$ in function of joints $\boldsymbol{q}_1$, $\boldsymbol{q}_2$, $\boldsymbol{q}_4$, $\boldsymbol{q}_5$ **and** $\boldsymbol{q}_6$

In this particular case, if the q_1, q_2, q_4, q_5 and q_6 joints are combined to the position vector functions (p_x, p_y, p_z), a vertical and horizontal path of the half sphere generated by the q_4, q_5 and q_6 joints is obtained through a half-circle path as shown below in Fig. 5.8.

Finally, by combining all the joints of the manipulator, a data set of size 244,140,625 data with 18 variables is obtained and generated through the functions of the position vector (p_x, p_y, p_z) as shown below in Fig. 5.9.

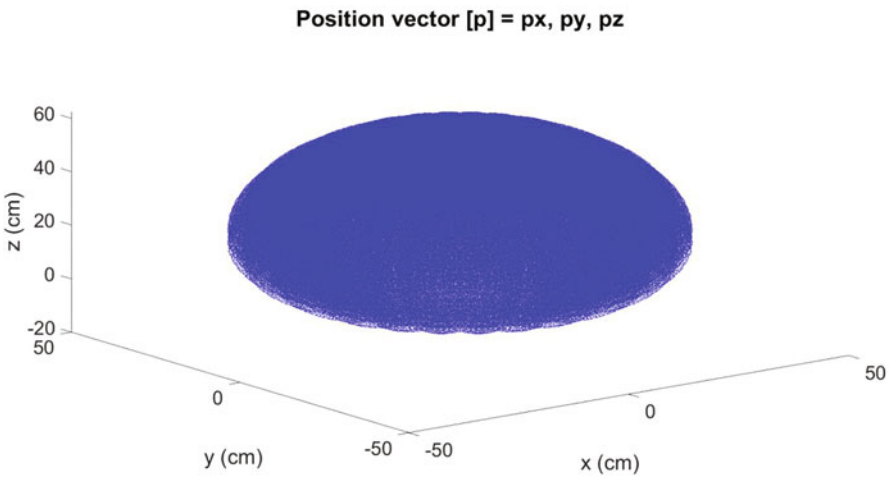


Fig. 5.9 Position vector (p_x, p_y, p_z) in function of joints q_1, q_2, q_3, q_4, q_5 **and** q_6

5.2.5 *Reduction Algorithm Based on Linear Systematic Sampling*

It is evident that the amount of data generated in set B exceeds the processing capacities of a conventional computer. To solve the problem, an algorithm based on the LSS method was designed. This method has two main advantages. On one hand, the selection of the first sample is chosen by a sampling period that guarantees a random instance among a set of samples in a given interval. On the other hand, the samples chosen consecutively are distributed in a good way among the population, that is, there is less risk that some or a large part of the population is not represented, maintaining a constant and uniform distribution among the data [4].

Although the LSS method is one of the most common methods, it has two drawbacks. On one hand, The sampling variance cannot be taken impartially on the basis of the single sample taken and in the other hand, when the population size N cannot be divided evenly by the desired sample size n, systematic sampling cannot be performed, that is, when N is not an integer multiple of the desired sample size n and consequently $N \neq nk$. In this case, the sampling could be inefficient and if at some point the characteristics of the population were periodic and they coincide with the sampling interval, the representativeness of the desired sample could be biased [10, 11].

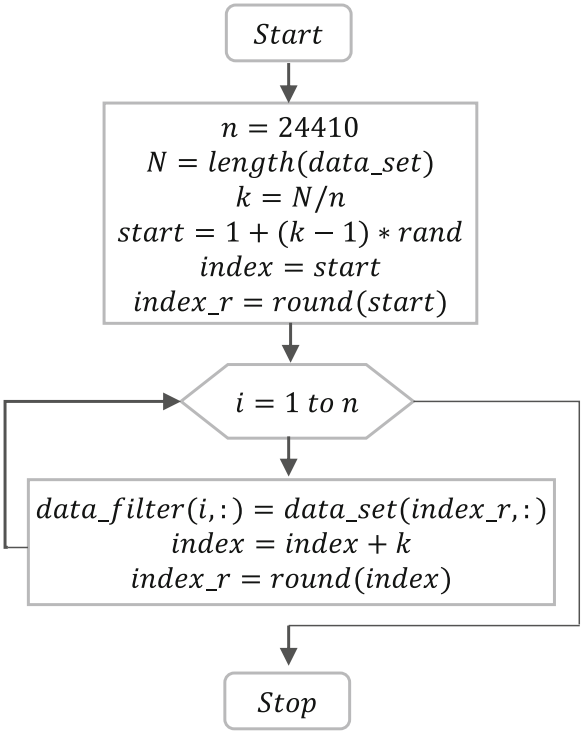
The quality of the data is influenced by the spatial resolution of the data in the workspace. In this sense, the higher the spatial resolution, the more data will be distributed in the workspace and the better the data quality. However, to carry out the processing on conventional processors, it will be necessary to implement data reduction algorithms that allow obtaining a balanced representation and thus obtaining good results in the knowledge extraction algorithms [4, 6].

In this case, considering data set B with a population of $N = 244{,}140{,}625$, where the size of the desired sample is $n = 24{,}000$, it is observed that N is not an integer multiple of n, because $k = N / n = 10{,}172.5260$; therefore, k is not an integer that satisfies the criteria of the LSS method. However, due to the distribution characteristics of the data, it is possible to carry out systematic sampling while maintaining a homogeneous representativeness among the data. In this case, the sampling interval k will be converted to an integer, because the characteristics of the population are not periodic; therefore, the accumulated bias at the end of the sampling can be eliminated without altering or modifying the homogeneous representativeness of the population. Figure 5.10 describes the RDF algorithm that was used only with data set B due to its size to fulfill this purpose.

According to the Fig. 5.10, the steps for the implementation of the filter are described in a general way:

1. The values for the original population size N and the desired sample size n are initialized.
2. The constant of the sampling interval $k = N / n$ is calculated.
3. The random number between 1 and k is generated.
4. The random value is rounded to an integer $r1$.
5. Element $r1$ from the original data set is selected as the first filtered data.

Fig. 5.10 RDF algorithm



6. The cumulative sum is made between the random number and the constant *k*. In this step, the cumulative sum is a positive noninteger value.
7. The cumulative value of the sum is rounded to an integer *r2*.
8. Element *r2* is selected from the original data set as the second filtered data and steps 6, 7, and 8 are repeated until the desired *n* samples are reached.

Where the variable *n* represents the desired sample, *N* represents the original population, *k* is the sampling interval, a random number between 1 and *k* is stored in the variable *start*, and the variable *index* represents the index of the instance with decimal number while *index_r* is the integer value of the index of the instance rounded up.

Within the iterative loop type for, the selection of instances is performed starting from the *k-th* element until reaching the desired sample *n*. The variable index accumulates the values of the indices with decimals and the value of the sampling interval *k*.

For the case of data set B, there is a population $N = 244{,}140{,}625$, where the desired sample is $n = 24{,}414$ and the sampling interval $k = 10{,}000{,}025$. In this case, when *k* is not an integer, the RDF algorithm will always perform rounding. Therefore, when the value of the modulus of the sampling interval *k* divided by the desired sample *n* is zero, a sampling without bias among the population is ensured, that is, when *K* is an integer, the LSS criterion is applicable. However, when the value of the modulus of the sampling interval divided between the desired samples is close to zero, the bias generated is minimal without significantly influencing the training results.

5.2.6 Data Set Normalization

The normalization of data sets A and B was performed in the range of −1 to 1, because the hyperbolic sigmoid tangent transfer function was used for the BPANN training. A mean of zero was applied, in an interval of $-1 \leq \text{data} \leq 1$ by means of Eq. (5.27).

$$\text{datanorm} = \left(\frac{\text{data-min}}{\text{range}} * (\text{high} - \text{low}) \right) + \text{low} \tag{5.27}$$

where *data_norm* is the normalized data, *data* is the data to normalize, *min* is the minimum of the generated data set, *range* is the range or difference between the maximum value and the minimum value of the generated data set, $high = 1$ represents the upper limit of the desired interval, and $low = -1$ represents the lower limit of the desired range [45].

5.2.7 Training and Test Data Sets

There is no procedure that determines the amount of training and test data that should be used to train ANNs. However, much of the research with ANN uses the 80:20 and 90:10 ratio. In this study, a random selection of the data was carried out in order to randomly choose the training and testing subsets with densities of 80:20 and 90:10. This procedure was applied to both data sets, *A* and *B*.

To analyze the performance in the proposed network architectures, a training was carried out for each proposed data density configuration. With the aim of analyzing the generalizability in each of the proposed network architectures, during the testing stage, an error of less than 5% was considered for the prediction of the data through a statistical analysis of correlation and Chi-square.

5.2.8 Training and Test Back Propagation Neural Network

To determine the optimal parameters of the BPNN network, the RDANN methodology based on the robust parameter design method proposed by Genichi Taguchi was used. The engineering method, applied to the design of products or processes, is focused on reducing the sensitivity to noise and has proven to be an efficient and powerful method in product design [46].

The robust design technique is known as factorial design of experiments where most of the possible combinations can be identified without the need to include a considerable number of experiments and its application allows determining the functionality or performance of a product or process to be controlled [47].

The RDANN methodology applied to the ANN design allows finding the selection of the factors involved that allow minimizing the variability of the response to different inputs to the system through the appropriate choice of the levels in the controlled design variables [19].

The design variables considered were the number of neurons in the first and second layers, the momentum, and the learning rate. For the noise variables, the random weights, the size of the training set versus the size of the test set, and the random selection of the sets were considered. During the experimentation stage, an orthogonal array configuration was used with an $L_9(3^4)$ y $L_4(3^2)$ with the aim of training and testing 36 different ANN architectures [48].

During the confirmation stage, the signal-to-noise ratio was analyzed through an analysis of variance (ANOVA) to determine adequate levels in the variables involved and to identify the possible optimal values of the best network topology, also involving the value of the mean obtained from the Mean Square Error (MSE). The best architecture was 12: 12: 6, momentum = 0.01, and learning rate = 0.1.

Figure 5.11 shows the generalization tests applied to the BPNN that was trained with 19,200 data from set A with spatial resolution of $10\times5\times5\times6\times4\times4$ without applying the filter. In each individual graph, the six joint coordinates of

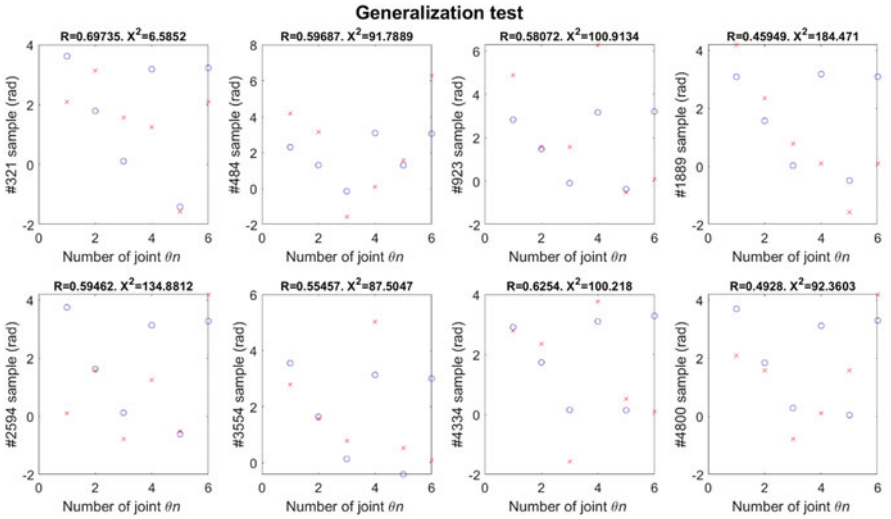


Fig. 5.11 Without filtering: Predicted joints vs calculated

the manipulator that were previously calculated ("x") versus the joint coordinates predicted by the BPNN ("o") are observed.

Figure 5.12 shows the tests carried out on the BPNN, with the data from set *B* that were previously treated by the RDF, with a spatial resolution of $25\times25\times25\times25\times25\times25$. Joint coordinates previously calculated and belonging to the test data set are displayed. In each box, six joint coordinates calculated ("x") versus joint coordinates predicted by the BPNN ("o") are observed.

5.2.9 Training and Test Generalized Regression Neural Network

In the GRNN, to determine the optimal spread value of the network, also known as the "kernel spread constant," a spread value smaller than the distance between the input vectors is used to make a close fit between the data. If a higher spread value is used, it could cause an over adjustment, on the contrary, if the value is too small, it could cause an under adjustment. Therefore, this constant propagation value is considered as a regularization parameter that should be optimally selected [49].

To determine the best constant propagation value in the RGNN, 2000 trainings were performed for each configuration in the data density (80:20 and 90:10) for each proposed set (A and B) in an automated manner. Next, Table 5.3 shows the spread values obtained during the RGNN training.

Figure 5.13 shows the generalization tests applied to the GRNN that was trained with 19,200 data from set A with spatial resolution of $10\times5\times5\times6\times4\times4$, without

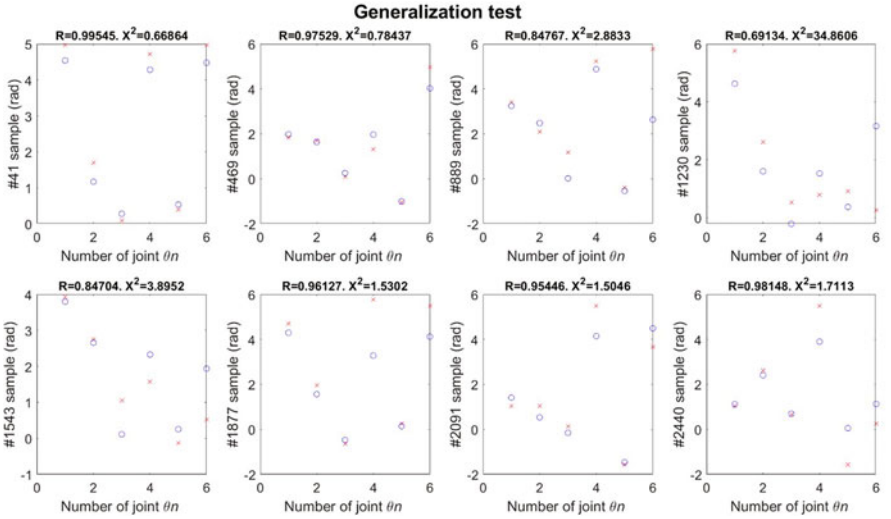


Fig. 5.12 With filtering: Predicted joints vs calculated

Table 5.3 Obtaining the spread value with 2000 iterations

Data set	Train/test	Global time [hr]	Optimal spread
A	80:20	2.38	0.2881
A	90:10	1.22	0.2721
B	80:20	2.26	0.2111
B	90:10	0.89	0.2131

applying the filter, with a much lower spatial resolution compared to set *B*. In each individual graph, the six joint coordinates of the manipulator that were previously calculated ("x") versus the joint coordinates predicted by the GRNN ("o") are observed.

Figure 5.14 shows the tests performed on the GRNN that was trained with the data from set B that were previously treated by the RDF, with a spatial resolution of 25x25x25x25x25x25. Eight joint coordinates previously calculated and belonging to the test data set are displayed. In each box, six joint coordinates calculated ("x") are observed versus joint coordinates predicted by the GRNN ("o") are observed.

5.3 Results

5.3.1 Reduction Data Filter Analysis

To analyze the distribution of the data, two dispersion matrices were plotted to compare the results obtained before and after applying the filter to the data. Based on the range of motion previously established for each of the joints in Fig. 5.15 (a), a

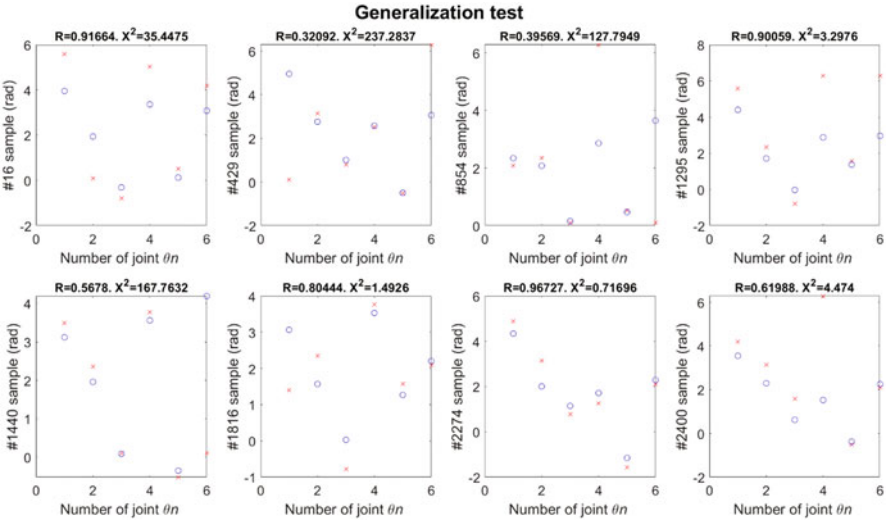


Fig. 5.13 Without filtering: Predicted joints vs calculated

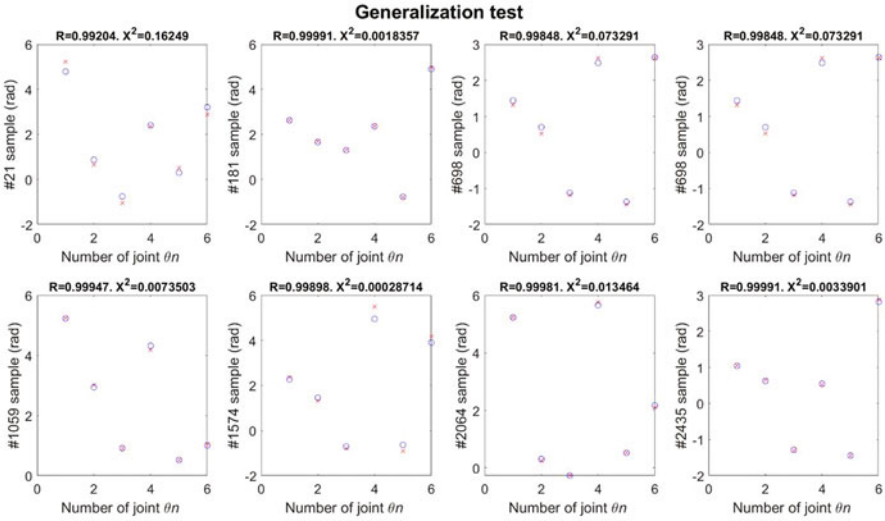


Fig. 5.14 With filtering: Predicted joints vs calculated

representation of data set B is shown, which maintains a population of 244,140,625 data for each variable. On the main diagonal, the distribution of three variables corresponding to the position vector $[p] = \{p_x, p_y, p_z\}$ and their data scatter diagrams in different perspectives can be observed.

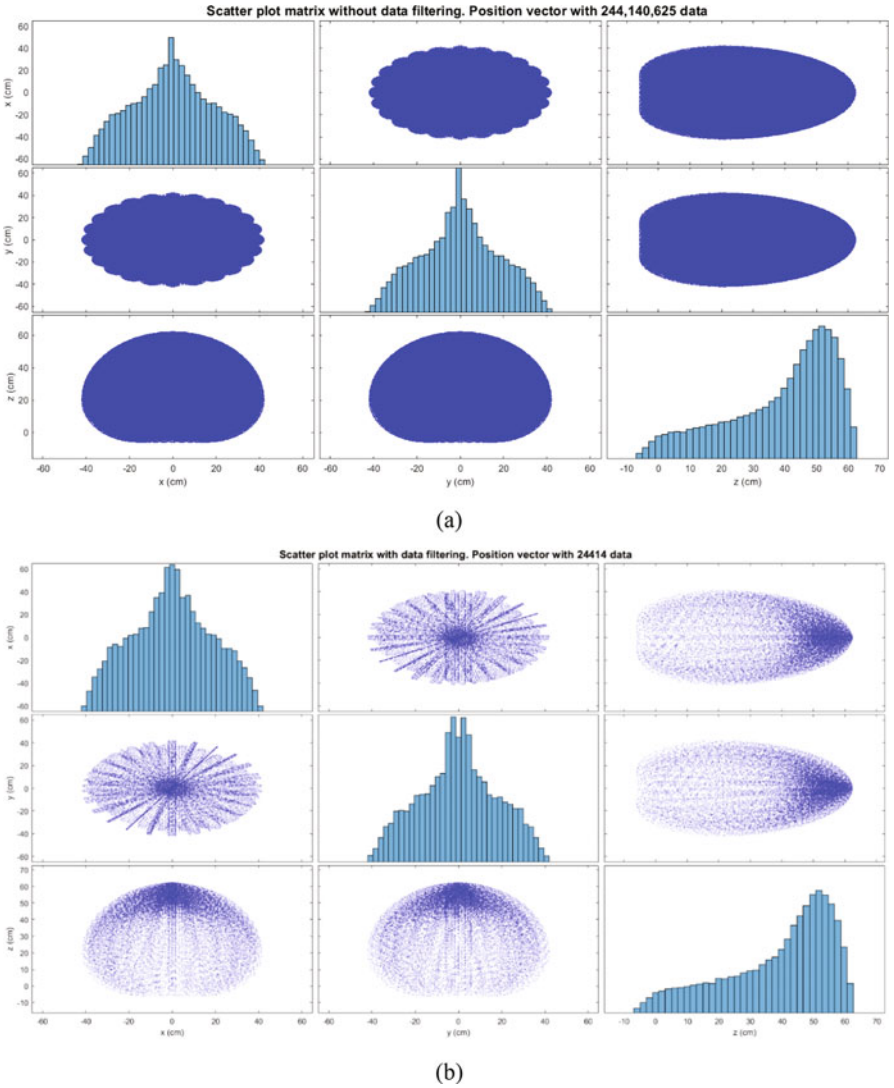


Fig. 5.15 Dispersion matrix of the position data set B: (**a**) Before filtering (**b**) After filtering

In Fig. 5.15 (b), the data after applying the RDF are shown, obtaining a reduction of $n = 24{,}410$ data for each variable. It can be seen that the data maintain a constant and uniform distribution with respect to the data set shown in Fig. 5.15 (a).

Figure 5.16 shows the distribution of the position and orientation data vectors before and after applying the RDF to the data. In Fig. 5.16 (a), we can see the distribution of the vectors of position $[p] = \{p_x, p_y, p_z\}$ and orientation $[n\ o\ a] = \{n_x,$

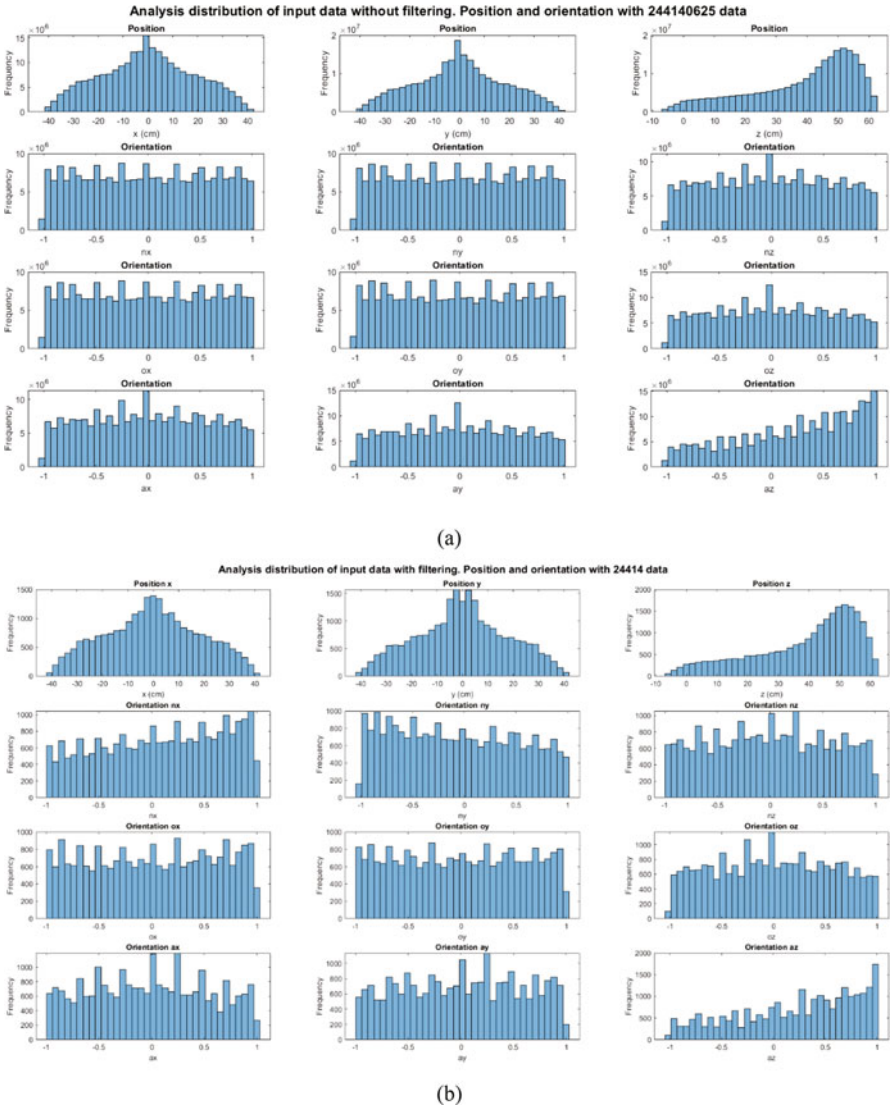


Fig. 5.16 Distribution of position and orientation data set *B*: (**a**) Before filtering (**b**) After filtering

n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z} belonging to data set B and which originally maintains a population of 244,140,625 data per variable.

In Fig. 5.16 (b), the data distribution after applying the RDF is shown, obtaining a reduction of $n = 24{,}410$ data per variable. It can be seen that most of the data maintains the distribution in a constant and uniform way with respect to the data set shown in Fig. 5.16 (a).

5.3.2 *Performance Evidence with Filtering: Comparison GRNN with BPNN*

Next, Table 5.4 shows the results obtained by the two network architectures in the testing stage, where it can be seen at the end of the table that RGNN obtained the best percentage of predictions with an error of less than 5%. Likewise, it can be observed that the training time required for this network is much lower compared to the BPNN.

The use of the RDF filter allowed solving the problem of data volume and also to carry out the training in a conventional processor, maintaining the homogeneous distribution of the original data. During the tests, an error of less than 5% was considered in the prediction of the data for both architectures by means of a statistical analysis of correlation and Chi square.

For the BPNN training, 10,000 iterations were performed and the *trainrp* training algorithm was used with a goal mse = 1E-4. The structural parameters of the network such as momentum and learning rate were obtained by the RDANN methodology with values of 0.01 and 0.1, respectively. The best results in this architecture were obtained through the training performed by data set B processed by the RDF Filter, with a data density of 90% for training data and 10% for tests.

5.4 Conclusions and Discussions

There is a relationship between the improvement in the generalizability of both proposed networks and the fact that the data set was previously processed by the RDF, because the reduction of the data allows maintaining a representative distribution with respect to the spatial resolution generated initially, maintaining the characteristics of the original population, and obtaining a considerable reduction of 99.99% to be able to carry out the processing in conventional computers and obtain good results.

For the application of the RDF algorithm, as long as the modulus of the sampling interval k divided by the desired sample n is a value close to zero, it will be possible to obtain a better representativeness of the population N. Therefore, the answer of question How close to zero should the sampling interval k divided by the desired sample n be? its determination will depend on the characteristics of the problem to consider it as a sampling that allows obtaining good results, since the expected results are not always obtained when considering an integer k sampling interval. In this study, the value was 0.02560.

The results obtained by the two network architectures showed an increase in the generalizability due to the fact that they were trained with the data set previously treated by the filter. However, when training both architectures using the data set that was not treated, the generalizability in both networks was lower.

Table 5.4 Comparative table of results obtained

				RGNN		BPNN	
Data set	Size before Filtering	Size after filtering	Train/Test	Training time [s]	%Rightguess $X^2 < 5\%$	Training time [s]	%Rightguess $X^2 < 5\%$
A	24,000 × 18	24,000 × 18	80:20	4.27	48%	264.41	44%
A	24,000 × 18	24,000 × 18	90:10	1.60	50%	316.02	46%
B	244,140,625 × 18	24,000 × 18	80:20	7.41	79%	175.44	71%
B	244,140,625 × 18	24,000 × 18	90:10	3.51	80%	184.84	72%
B	244,140,625 × 18	24,414 × 18	80:20	2.69	83%	189.04	73%
B	*244,140,625 × 18*	*24,414 × 18*	*90:10*	*1.58*	*83%*	*158.57*	*74%*

The increase in the percentage of successes in relation to the spatial resolution in the training data sets is evident. However, as spatial resolution increases, data processing requires higher performance in the available hardware resources, so the benefit that the use of data preprocessing techniques and/or tools that allow obtaining good results without decreasing performance and generalizability in knowledge extraction algorithms is evident.

The time required to train a BPNN is usually ten times greater than the training time required by a GRNN, in addition to the fact that it is not necessary to select a list of structural parameters compared to the BPNN, so its implementation is faster and more efficient.

The best training in both network architectures was obtained using a data density of 80% for training and 20% for testing. At best, during the testing phase, the GRNN succeeded in keeping 83% of its predictions with a margin of error of less than 5%. Therefore, this study reaffirms that the quality of the training data sets has a significant influence on the results obtained by the knowledge extraction algorithms.

Acknowledgments This work was supported by IPN-COTEBAL to study doctoral degree under permissions 14/2018, 33/2019, and 23/2020. This works was partially supported by CONACYT - BecasNacionales de Posgrado con la Industria under contract 431101/640582, by the collaborate students Agustín Ortíz and Brayan M. Carrera and by OMADS S.A. of C.V., an enterprise dedicated to the innovation and technological development.

References

1. C. Marco, P. Anit, How organisations leverage big data: A maturity model. Ind. Manag. Data Syst. **116**(8), 1468–1492 (2016). https://doi.org/10.1108/IMDS-12-2015-0495
2. K.-C. Li et al., *Big Data: Algorithms, Analytics, and Applications* (Chapman and Hall/CRC, 2015)
3. H. Liu, H. Motoda. On issues of instance selection. Data Min. Knowl. Disc. **6**(2) Art. no. 2 (2002). https://doi.org/10.1023/A:1014056429969
4. H. Liu, H. Motoda, Instance Selection and Construction for Data Mining. Springer US (2001)
5. S. García, J. Luengo, F. Herrera, *Data Preprocessing in Data Mining*, vol 72 (Springer International Publishing, Cham, 2015)
6. B. Gu, F. Hu, H. Liu. Sampling and its application in data mining. Technical Report TRA6/00, Department of Computer Science, National University of Singapur (2000)
7. H. Liu, H. Motoda, On issues of instance selection. Data Min. Knowl. Disc. **6**(2), 115–130 (2002). https://doi.org/10.1023/A:1014056429969
8. H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms. Data Min. Knowl. Disc. **6**(2), 153–172 (2002). https://doi.org/10.1023/A:1014043630878
9. W. G. Madow, L. H. Madow. On the theory of systematic sampling, I. Ann. Math. Stat. **15**(1). Art. no. 1 (1944)
10. S.A. Mostafa, I.A. Ahmad, Remainder linear systematic sampling with multiple random starts. J. Statist. Theory Pract. **10**(4), 824–851 (2016). https://doi.org/10.1080/15598608.2016.1231094
11. L.H. Madow, Systematic sampling and its relation to other sampling designs. J. Am. Stat. Assoc. **41**(234), 204–217 (1946). https://doi.org/10.1080/01621459.1946.10501864
12. A. Serrano García. Inteligencia artificial (2016)

13. X. He, S. Xu, SpringerLink (Online service), *Process Neural Networks: Theory and Applications* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010)
14. A.K. Jain, J. Mao, K.M. Mohiuddin, Artificial neural networks: A tutorial. Computer **29**(3), 31–44 (1996). https://doi.org/10.1109/2.485891
15. L. Aggarwal, K. Aggarwal, R.J. Urbanic, Use of artificial neural networks for the development of an inverse kinematic solution and visual identification of singularity zone(s). Procedia CIRP **17**, 812–817 (2014). https://doi.org/10.1016/j.procir.2014.01.107
16. J. Zupan, *Introduction to Artificial Neural Network (ANN) Methods: What They Are and How to Use Them*, vol 41 (1994)
17. Y. Zhang, D. Guo, Z. Li, Common nature of learning between back-propagation and hopfield-type neural networks for generalized matrix inversion with simplified models. IEEE Trans Neural Netw. Learn. Syst. **24**(4), 579–592 (2013). https://doi.org/10.1109/TNNLS.2013.2238555
18. T. Ozaki, T. Suzuki, T. Furuhashi, S. Okuma, Y. Uchikawa, Trajectory control of robotic manipulators using neural networks. IEEE Trans. Ind. Electron. **38**(3), 195–202 (1991). https://doi.org/10.1109/41.87587
19. J.M. Ortiz, R. del Martínez, J.M.C. Viramontes, H.R. Vega, *Robust Design of Artificial Neural Networks Methodology in Neutron Spectrometry* (Artificial Neural Networks - Architectures and Applications, 2013). https://doi.org/10.5772/51274
20. Specht, Probabilistic neural networks for classification, mapping, or associative memory, in *IEEE 1988 International Conference on Neural Networks*, 1988, pp. 525–532 vol.1, doi: https://doi.org/10.1109/ICNN.1988.23887
21. D.F. Specht, Probabilistic neural networks. Neural Netw. **3**(1), 109–118 (1990). https://doi.org/10.1016/0893-6080(90)90049-q
22. D.F. Specht, P.D. Shapiro, Generalization accuracy of probabilistic neural networks compared with backpropagation networks. IJCNN-91-Seattle Int. J. Conf. Neural. Netw. **i**, 887–892 (1991). https://doi.org/10.1109/IJCNN.1991.155296
23. P. Jha, B. B. Biswal. A neural network approach for inverse kinematic of a SCARA manipulator. *IAES International Journal of Robotics and Automation*, **3**(1), Art. no. 1 (2014). https://doi.org/10.11591/ijra.v3i1.3201
24. Lee, Robot arm kinematics, dynamics, and control. Computer **15**(12), 62–80 (1982). https://doi.org/10.1109/MC.1982.1653917
25. B. Karlik, S. Aydin, An improved approach to the solution of inverse kinematics problems for robot manipulators, Eng. Appl. Artif. Intell. **13**(2), Art. no. 2, (2000). https://doi.org/10.1016/S0952-1976(99)00050-0
26. L. Jin, S. Li, J. Yu, J. He, Robot manipulator control using neural networks: A survey. Neurocomputing **285**, 23–34 (2018). https://doi.org/10.1016/j.neucom.2018.01.002
27. S. Li, Y. Zhang, L. Jin, Kinematic control of redundant manipulators using neural networks. IEEE Transact Neural Netw Learn Syst **28**(10), 2243–2254 (2017). https://doi.org/10.1109/TNNLS.2016.2574363
28. M. Tarokh, M. Kim, Inverse kinematics of 7-DOF robots and limbs by decomposition and approximation. IEEE Trans. Robot. **23**(3), 595–600 (2007). https://doi.org/10.1109/TRO.2007.898983
29. R. Köker, T. Çakar, Y. Sari, A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators. Eng. Comput. **30**(4), 641–649 (2014). https://doi.org/10.1007/s00366-013-0313-2
30. B.K. Bose, Neural network applications in power electronics and motor drives—An introduction and perspective. IEEE Trans. Ind. Electron. **54**(1), 14–33 (2007). https://doi.org/10.1109/TIE.2006.888683
31. A. Hasan, A.T. Hasan, H.M.A.A. Al-Assadi, Performance prediction network for serial manipulators inverse kinematics solution passing through singular configurations. Int. J. Adv. Robot. Syst. **7**(4), 11–24 (2011)
32. R. Gao. Inverse kinematics solution of Robotics based on neural network algorithms. J. Ambient Intell. Humanized Comput. (2020) https://doi.org/10.1007/s12652-020-01815-4

33. L. Jin, S. Li, J. Yu, J. He, Robot manipulator control using neural networks: A survey. Neurocomputing **285**, 23–34 (2018). https://doi.org/10.1016/j.neucom.2018.01.002
34. A.-M. Zou, Z.-G. Hou, S.-Y. Fu, and M. Tan. Neural Networks for Mobile Robot Navigation: A Survey. In *Advances in Neural Networks - ISNN 2006* (2006), pp. 1218–1226
35. X. Wu, Z. Xie, Forward kinematics analysis of a novel 3-DOF parallel manipulator. Scientia Iranica. Transact. B Mechan. Engin. **26**(1), 346–357 (2019). https://doi.org/10.24200/sci.2018.20740
36. V. Khoshdel, A. Akbarzadeh, Eds. An optimized artificial neural network for human-force estimation: consequences for rehabilitation robotics. Industr. Robot. Intern. J. **45**(3), Art. no. 3 (2018). https://doi.org/10.1108/IR-10-2017-0190
37. R. Fernando, Robótica – control de robots manipuladores. Alfaomega Grupo Editor (2011)
38. R. Köker, Reliability-based approach to the inverse kinematics solution of robots using Elman's networks. Eng. Appl. Artif. Intell. **18**(6), 685–693 (2005). https://doi.org/10.1016/j.engappai.2005.01.004
39. A. Larrañaga. 3D Printable Robotic Arm. GitHub (2018). https://github.com/AngelLM . Accessed 18 Sep 2019
40. R. Köker, C. Öz, T. Çakar, H. Ekiz, A study of neural network based inverse kinematics solution for a three-joint robot. Robot. Auton. Syst. **49**(3), 227–234 (2004). https://doi.org/10.1016/j.robot.2004.09.010
41. S. Tejomurtula, S. Kak, Inverse kinematics in robotics using neural networks. Inf. Sci. **116**(2), 147–164 (1999). https://doi.org/10.1016/S0020-0255(98)10098-1
42. J. Denavit, R.S. Hartenberg, A kinematic notation for lower-pair mechanisms based on matrices. Trans. ASME J. Appl. Mech. **22**, 215–221 (1955)
43. A.R.J. Almusawi, L.C. Dülger, S. Kapucu, A new artificial neural network approach in solving inverse kinematics of robotic arm (Denso VP6242). Comput Intell Neurosci CIN **2016** (2016). https://doi.org/10.1155/2016/5720163
44. S. García, J. Luengo, F. Herrera, *Data Preprocessing in Data Mining*, vol 72 (Springer International Publishing, Cham, 2015)
45. T. Jayalakshmi, A. Santhakumaran, Statistical normalization and Back propagation for classification. Intern. J. Comput. Theory Eng. **3**(1), 89–93 (2011)
46. J. Limon-Romero, D. Tlapa, Y. Baez-Lopez, A. Maldonado-Macias, L. Rivera-Cadavid, Application of the Taguchi method to improve a medical device cutting process. Int. J. Adv. Manuf. Technol. **87**(9–12), 3569–3577 (2016). https://doi.org/10.1007/s00170-016-8623-3
47. M. Ibrahim, N. Zulikha, Z. Abidin, N.R. Roshidi, N.A. Rejab, M.F. Johari, Design of an Artificial Neural Network Pattern Recognition Scheme Using Full Factorial Experiment. Appl. Mech. Mater. **465–466**, 1149–1154 (2013). https://doi.org/10.4028/www.scientific.net/AMM.465-466.1149
48. T. Y. Lin, C. H. Tseng. Optimum design for artificial neural networks: an example in a bicycle derailleur system. Eng. Appl. Artif. Intell. **13**(1), Art. no. 1 (2000). https://doi.org/10.1016/S0952-1976(99)00045-7
49. D.-S. Huang, Radial basis probabilistic neural networks: model and application. Int. J. Pattern Recognit. Artif. Intell. **13**(7), 1083–1101 (1999). https://doi.org/10.1142/S0218001499000604