IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 10, 2020, accepted December 26, 2020, date of publication January 1, 2021, date of current version January 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3048877

Online Measuring of Robot Positions Using Inertial Measurement Units, Sensor Fusion and Artificial Intelligence

BENEDITO A. N. CAMPOS AND JOSÉ MAURÍCIO S. T. MOTTA

Department of Mechanical and Mechatronics Engineering, University of Brasilia—Darcy Ribeiro University Campus, Brasília 70910-900, Brazil

Corresponding authors: Benedito A. N. Campos (beneditocampos@b2ml.com.br) and José Maurício Motta (jmmotta@unb.br)

ABSTRACT This research introduces a new method to estimate the position of a robot's Tool Center Point (TCP) using Inertial Measurement Units (IMUs), sensor fusion and Artificial Neural Networks (ANNs). The objective is to make an accurate estimate of TCP navigation, using the signals from an IMU as resources of a neural network capable of predicting the position. Considering that the IMU sensors suffer noise in the measurements and the noise progresses over time, this proposal employs a technique that eliminates the filtering step, and the process is done internally by the network. The work employs a non-parametric approach to reset the reference dynamically, minimize noise from sensors, and converge positioning to a nominal result. This method offers a solution for fast, cheap, and efficient robot calibration. The work does not want to replace current techniques but to introduce a new design to the literature. The concept does not require sophisticated mechanical parts and the production line to be idle during the calibration process, and the results show that the developed technique can accurately predict the TCP position with millimeter errors and in real-time. The study also implemented the concept with other neural networks, for which it used a smaller set of data in an attempt to reduce training time. The research used the Multilayer Perceptron and XGBRegressor networks to test the approach introduced with others algorithms. Different applications that need real-time positioning can benefit from the proposal.

INDEX TERMS IMU calibration, neural networks, online robot calibration, sensor fusion.

I. INTRODUCTION

Robots today play a central role in the manufacturing industry [1], [2], and for these machines to work correctly it is necessary to perform their calibration. Robot calibration is a systemic process of modeling, measuring, numerically identifying its physical characteristics and implementing a new model [3]–[5]. Thus, kinematic calibration is a way to improve the positioning accuracy of the robot [6]. In the case of industrial robots, it is a method of minimizing the effects of various sources of errors that affect the position and orientation (pose) accuracy of the robot TCP, due to geometric deviations and other sources of errors during its operation [7].

In the process of robot calibration, the robot's TCP poses are measured [8] and the deviations between the desired poses in the robot program and those reached in its operation are recorded. Then, complex nonlinear numerical techniques generally employ a function capable of adjusting the kinematic model and its parameters according to the measured position [9], but for this, it is necessary to use sophisticated measuring devices [10] and interrupt the machine's operation. There are also non-parametric techniques [11], using regression equations and neural networks [12] to perform this procedure. All processes require measuring instruments with high precision and, therefore, costly for the industry. The aim of the research is to employ sensors and neural networks to perform the estimated TCP navigation (Dead Reckoning) [13], reducing the complexity of external measurement systems.

A. MOTIVATION

To develop a robot calibration method, it is necessary to measure the actual operating position and compare it with the desired nominal position and then adjust the kinematic model of the robot to compensate for deviations. This work introduces a technique for mapping the robot's TCP position

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Jiang.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

using Inertial Measurement Units (IMUs), and Artificial Neural Networks (ANNs) as a software tool capable of mitigating the progressive and inherent noise and error propagation of the IMU sensors, thus predicting its real position in space. Regarding this processes, there are two a priori approaches. The first is the parametric one [14], [15], in which the error is related to parameters that reflect geometry and other mechanical characteristics such as elasticity, eccentricity, clearance, and others. The second approach is known as non-parametric [12], where the robot model is submitted to a nonlinear regression equation or other methods, such as the case of neural networks [12], [16], [17].

As for the practical aspect there are two types of robot calibration processes: the offline calibration process and the online one. In general, in the offline process the robot remains out of work and a calibration program performs the process. The algorithm guides the robot TCP to selected positions, then tools capable of measuring the actual position are employed. Next, the error is calculated from measured points and the desired nominal coordinates. In this case, the classic calibration method involves the development of a kinematic model with error parameters that is supposed to accurately describe the real robot position [18], [19]. In the online calibration process, the kinematic model in the controller is adjusted while the robot operates, without the need to suspend its operation to perform the calibration due to natural wear and maintenance stops or tool changes. In this case, it is necessary to estimate the robot's TCP position while the robot operates, using a sensing device that is sufficiently accurate and fast. From an economic point of view, it is not necessary to interrupt the service of the robot on the manufacturing line to calibrate it.

Despite the little use of robot calibration methods online, the development of Micro Electromechanical Systems (MEMs) [20]–[22], which has risen the interest of researchers in various areas over time [23], has now encouraged research in robot calibration with IMUs with gyroscopes and accelerometers coupled to measure the robot's TCP orientation at the programmed points [24], [25]. These inertial devices, which used to be large and expensive and were used mostly in the aerospace industry, are already available at prices below one hundred dollars and weighing around twenty grams. Thus, methods based on IMUs could remove the complexity of the operation in terms of hardware, such as measuring arms, interferometry systems, laser tracking, and others. Current literature discusses the use of IMUs only to calibrate the orientation of the robots [24], [25], excluding position because IMU systems suffer from the fast accumulation of errors during the measurement process, and this makes it impossible to measure the position. However, it is feasible to compensate for such errors through software techniques.

Position and orientation can be obtained by integrating the measurements made by the inertial sensor, but these estimates are not accurate over time due to progressive noise. To mitigate this situation, it is often essential to fuse sensors with other devices [26]. Since sensors can provide estimates with high sampling rate (accurate on a short time scale), it is necessary to combine this method with sensors that have a lower sampling rate [26]. This research aims to eliminate this arrangement of devices, using only IMU signals to feed the neural network and predict position in space during robot displacement.

IMUs are composed of sets of inertial sensors, such as gyroscopes and accelerometers [27], [28], which provide rotation speed and linear acceleration along three directions of the a body structure. There are several types of gyroscopes, and those based on MEMs [29] are the selected type for this study. They are commercially available, accessible, and small [29]. However, some types of gyroscopes are limited to military use due to their high performance or price [30]. Accelerometers are an integral part of inertial navigation systems and they measure the acceleration of a body, indirectly, by measuring the specific force in the body frame of the IMU. Like gyroscopes, accelerometers have evolved in terms of price, energy consumption and resistance to the miniaturization characteristics observed in MEMs [31].

B. SENSOR FUSION

Many factors influence the error in an IMU, including instability in the gyroscope bias, uncertainty of the accelerometer, scale factor, mechanization in the calculation of integrals, non-linearities, and others [32]. Studies show that due to inconsistencies the longer a gyroscope operates the higher the stored rate or the error in the Angular Random Walk position and, since noise has a high-frequency element (short term) and low-frequency component (long term), if noise is not limited, the accuracy of the system will decrease [32], [33].

Researchers have concluded that adding other signal sources to the IMU would improve its performance [34]. Such devices are integrated on the same circuit board that contains MEMS-based IMUs to provide additional accuracy to the entire system [32]. A microprocessor calculates, integrates and combines the independent measurements of each device and thus compensates the deviations.

This research uses noisy sensor data to feed an ANN and get the actual robot TCP position along the guided trajectory. The goal is not to filter the IMU signals but to use them directly so that this step is eliminated. The filtering process is done by the neural network, which during training correlates the input to the expected output, thus being able to identify the characteristic noise of the IMU signals and then remove it. The noise removal process is implicit, avoiding the use of additional filtering methods. The motivation is to use noisy signals to predict a trajectory in 3D space, making unnecessary to filter them a priori, which eliminates the use of expensive IMUs and sophisticated filtering mechanisms.

C. ARTIFICIAL NEURAL NETWORKS

An Artificial Neural Networks is a circuit composed of a large number of simple processing units (neurons) inspired by a natural neural network [35] and is a massively parallel and distributed system where these simple units can

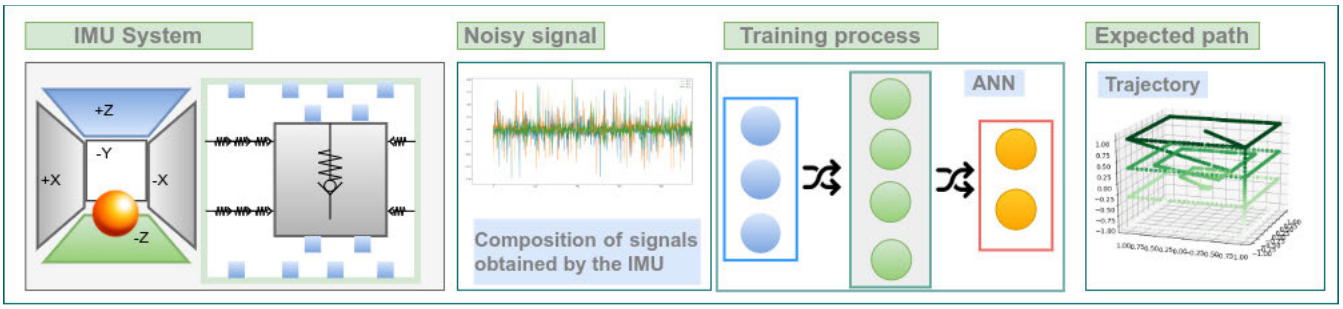

FIGURE 1. The IMU acquires the *x*, *y*, and *z* point coordinates using sensor fusion. Then the signal contaminated with noise is used to feed the ANN, which mitigates errors and predicts the trajectory of the guided movement.

store and use the knowledge [35], [36]. The learning process can be understood as the variation of synaptic weights over time [37], depending on the inputs it receives, until it stabilizes and reflects a correct relationship between a set of inputs (domain) and its image, representing something significant like the pattern recognition, the approximation of a function, a type of regression and others.

Considering that the ANN can emulate a non-linear regression method capable of mitigating noise from sensors through the correlation of variables, it is feasible to develop a system that predicts the robot TCP position based on the data sampled by the micro-sensor. The work describes the development of a system capable of predicting the position of a robot TCP in real-time, using the signals from the IMU and a neural network. Figure 1 generalizes the concept presented.

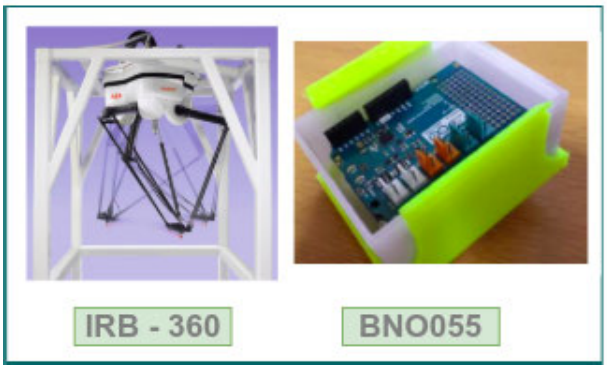

FIGURE 2. Robot and IMU employed. The IMU was placed on the tip of the robot.

II. MATERIALS AND METHODS

A. IMU AND ROBOT

In this research, an IRB-360 robot was used to perform a guided movement. The IMU used was the BNO055, as it is accessible (no controlled sale), cheap, and has a sensor fusion algorithm that allows configuration. The IMU was attached to the robot TCP. Figure 2 shows the components mentioned. The IMU provided 19 outputs to the network, that is, features were obtained from the fusion of sensor signals that the hardware made available and are as follows:

- Three inputs related to orientation *(x, y, z)* based on a 360-degree sphere;
- Angular velocity vector with three axes *(x, y, z)* represented for three features;
- Three characteristics referring to the angular acceleration vector in the three axes *(x, y, z)*;
- Magnetic field in the three axes *(x, y, z)* represented by three inputs;
- Three features are related to linear acceleration in the three axes *(x, y, z)*;
- Gravity in the three axes *(x, y, z)*;
- The temperature in degrees Celsius.

B. TRAJECTORY SETS

The program generated several sets of trajectories for the robot, and the coupled IMU produced the measured samples to the algorithm. As the ANNs are sensitive to information, the experiment provided the network with a large amount of data in the first experiment, so 6.2 million samples were collected. The paths that the method considered are composed of circles, squares, and ramps. The circles were programmed in two directions (clockwise and counterclockwise) in the planes (x, y) and (z, x), representing 30% of the samples. The square trajectories are also in both directions (clockwise and counterclockwise), but in this case, only in the plane (x, y) and describe 30% of the sample set. The ramps are in the three dimensions (x, y, z) and are responsible for 40% of the trajectories. Figure 3 exemplifies the generation of trajectories by software.

C. DATA PREPARATION

As discussed earlier, the measurements taken by the sensors are not accurate, and noise tends to increase over time. One way to place the values within the same range is through the technique called Feature Scaling [38], which facilitates the operation of most neural networks (normalize independent variables on a specific scale. As the input is noisy, this implies that the range of raw values varies widely and, therefore, hampers the objective function of many networks to function correctly. Any feature that has a large range of values will have greater weights in the training process. The objective of normalization is to ensure that the features are on the

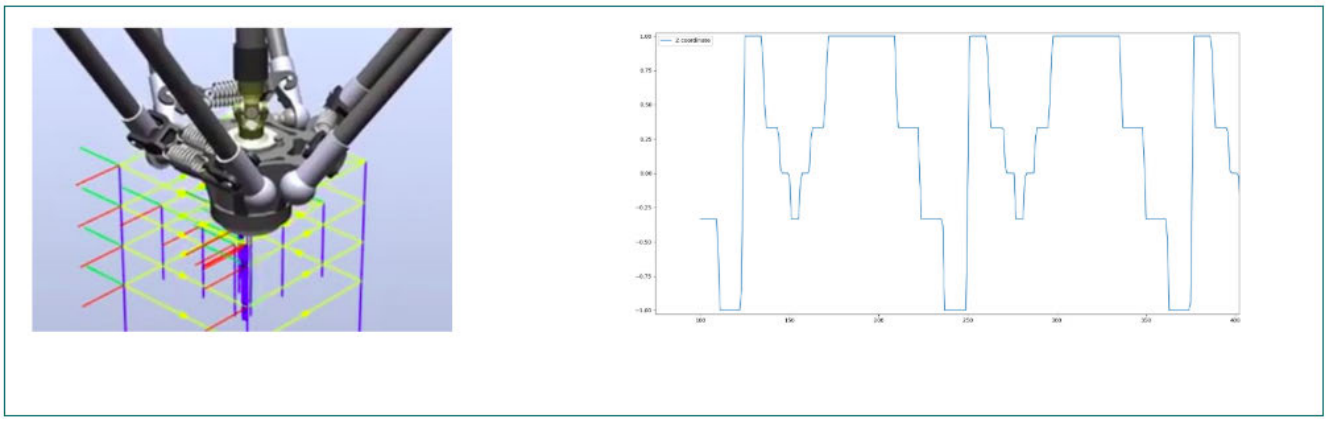

FIGURE 3. Example of the scheme used to generate the trajectories of the robot.

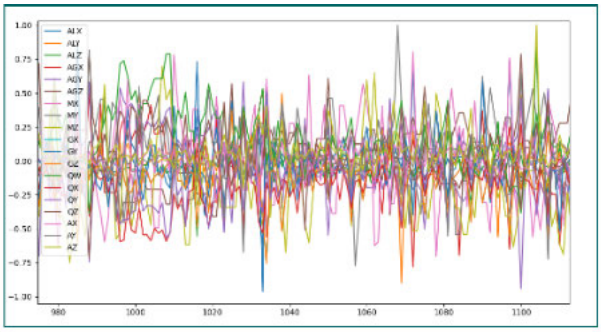

FIGURE 4. The 19 features delivered by the fusion of IMU sensors and their respective normalization for the convergence process.

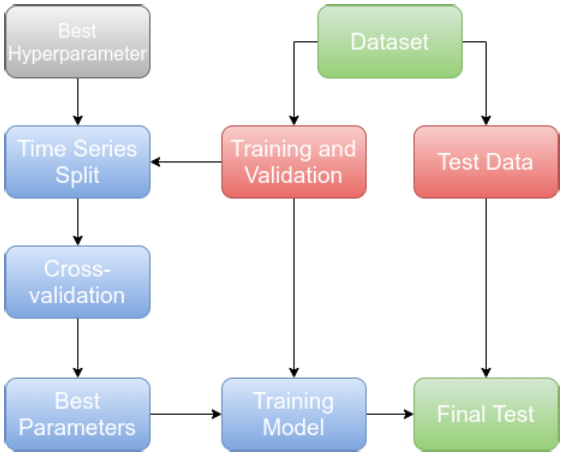

FIGURE 5. Scheme of data division, cross-validation and testing.

same scale of values, so there is no distortion between the intervals of the inputs and the neural network processes all information in a homogeneous way; therefore, despite the variation, the system can better correlate data to specify the model. Besides, the gradient method converges much faster using normalization [39]. Figure 4 shows normalization using the range $(1, -1)$.

D. TRAINING PROCESS

During training, two unwanted situations can occur, over-training and under-training [40]. The first case is known as excessive training and happens when the network obtains a specific solution. Therefore, the regression system is not able to predict values outside a known series. In the second case, the network is insufficiently trained, which leads to poor resolution, even using known samples. To verify that the network is not under-trained or over-trained, it is necessary to divide the data set into three partitions, respecting a defined percentage. The first partition is for training, used by the algorithm in the adjustment process. Then, a cross-validation protocol (suitable for time series) uses the validation set to check the quality of the results. Finally, the programmer observes the test data to evaluate the results and compare them with the expected output.

As there is a temporal relationship between the samples, it is not possible to randomize the data and then divide them into a training and test set (random division or K-Fold may not deliver good results). The data are treated as a time series, and therefore the methodology employed a validation technique known as continuous cross-validation, which returns the first K-folds as a training set and the $(k + 1)$th fold as a test set. The method is a variation of the K-Fold suitable for data sets that have a temporal relationship. The methodology trained and validated the ANN-RB network in a data set with different trajectories and varied directions and then tested the model obtained in the test set. The other algorithms were trained and evaluated in a series with 3825 samples and tested with 4500 entries. Figure 5 shows the general design used in the data preparation stage. Figure 6 illustrates how the methodology organized the training and validation set splits.

- **Training and Validation:** 85% of the data in the training and validation process with 6 splits. This procedure was applied to both algorithms (respecting the number of samples in each set).
- **Test:** This data is sent to the network after training as a necessary step to check if the model is robust. The survey

FIGURE 6. Time Series cross-validation used in the preparation of data for training.

considered 15% of the values measured in the test step for the ANN-RB and employed 4500 samples to test the others regressors.

The research employed two criteria to evaluate the model. The first is based on the construction of a Multi-layer Perceptron (MLP) network, where we defined its parameters and created the mathematical rules of the system. The second test battery used third-party tools and a smaller data set to try to get a solution that is not as dependent on the amount of data.

E. PROPOSED ANN

The developed network is based on the MLP algorithm with back-propagation, as it guarantees the convergence of the model and will be named hereafter ANN-RB network. Its implementation can be parallel and, thus, allows fast calculation using high sampling frequencies; the algorithm allows acceleration using of Nvidia's graphics processing units. For this, the algorithm was developed in the C++ language and employed the following steps in its construction:

1) The ANN receives input signals, $x_1, x_2, \ldots, x_{19}$, from IMU;
2) Existing weights (random at the beginning) w_1, $w_2, \ldots, w_{19}$ multiply the input signals;
3) The algorithm applies an activation function to the output: threshold, sigmoid or linear, creating the output signal y of this neuron;
4) The output y is then compared to the desired y_d (expected output): algorithm supervision factor;
5) The network calculates the error, which is the difference between the estimated value y and the expected value y_d;
6) The adaptation process takes place, which in this case is the change of the synaptic weights of the ANN neurons in the function of the error signal received and the derivatives in each synapse of the network;
7) After many passes of the sign, the weights w_i are altered and end up describing something relevant containing a small associated error.

Mathematically, the value y calculated at the output of each neuron depends on the input x, the weights w, and the activation function $f(.)$ that is applied to the weighted sum, according to equation (1).

$$y_{(i)} = f\left(\sum_{k=1}^{m}(w_k(i) * x_k(i))\right) \tag{1}$$

The error is the difference between this calculated value and the desired value, known in the model output, described by (2).

$$e_{(i)} = d_{(i)} - y_{(i)} \tag{2}$$

The learning process takes place by updating the weights of neurons in each iteration. The objective is to obtain a set of values that reduces the error between the output calculated by the network and the actual output. The literature shows that considering the objective of minimizing the error it is interesting to apply an energy function to the error. Since $E_{(w)}$ is a function of the energy applied to the weight vector w, which is continuously differentiable, then there is a $E_{(w*)}$ that makes the values w * optimal. The relationship between them is given by equation (3).

$$E(w*) \leqslant E(w) \tag{3}$$

The necessary condition to minimize the function $E(w*)$ is:

$$\nabla E(w*) = 0 \tag{4}$$

where ∇ is the gradient operator:

$$\nabla = \left[\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, \frac{\partial}{\partial w_3} \ldots \frac{\partial}{\partial w_n}\right]^T \tag{5}$$

The gradient of the cost function is:

$$\nabla E(W) = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \frac{\partial E}{\partial w_3} \ldots \frac{\partial E}{\partial w_n}\right]^T \tag{6}$$

The algorithm adopts a descending iteration, where at each step the cost function decreases as in (7). Initially, the algorithm starts with an initial estimate denoted by $w(0)$ and generates the weight vector sequence $w(1), w(2), \ldots, w(n)$. In equation (7), $w(n)$ represents the old value of the weight vector, and $w(n+1)$ is the new value. According to S. Haykin [36], the function will decrease based on the difference calculated between the old and the new weight vector, which is obtained by updating the weights.

$$E(w(n+1)) < E(w(n)) \tag{7}$$

Adjusting the weights at each iteration is performed in the direction of the steepest descent (against the gradient vector). Equation (8) describes the steepest descent algorithm as:

$$w(n+1) = w(n) - \eta \nabla E(w) \tag{8}$$

In (8), η is the learning rate (positive constant), known as step size, and $\nabla E(w)$ is the gradient vector considered at point $w(n)$.

The last layer of the network has its value compared to the final error, and its error signals are back-propagated to modify

the weights. In ANN we can define an error signal e in neuron j during a iteration n as equation (9):

$$e_j(n) = d_j(n) - y_j(n) \tag{9}$$

Thus the instantaneous energy of the error of the neuron j in the iteration n is:

$$E_{instantaneous} = \frac{1}{2}e_j^2(n) \tag{10}$$

The total energy obtained by adding the energy of all neurons in the output layer in the n iteration would be defined by equation (11):

$$E(n) = \frac{1}{2}\sum_{j\in c} e_j^2(n) \tag{11}$$

Since N is the sum of all iterations n, the total average energy will be:

$$E_{average} = \frac{1}{N}(\sum_{K=1}^{N} E(n)) \tag{12}$$

Here, average as well as instantaneous energy are functions of all free parameters such as synaptic weights and bias. Setting v as the weighted sum of each neuron, according to equation (13):

$$v_j(n) = \sum_{i=0}^{m}(w_{ij}(n) * y_{ki}(n)) \tag{13}$$

In equation (13), m is the total number of inputs that reach neuron j. The bias applied to neuron j is equal to the synaptic weight w_{j0}, so the output $y_j(n)$ of each neuron j corresponding to iteration n is described in (14).

$$y_j(n) = \varphi_j(v_j(n)) \tag{14}$$

The correction $\nabla w_{ij}(n)$ in the synaptic weight $w_{ji}(n)$ applied by the back-propagation algorithm is proportional to the partial derivatives $\frac{\partial E(n)}{\partial w_{ji}(n)}$, and according to the chain rule (calculus) the gradient can be defined as [36]:

$$\frac{\partial E(n)}{\partial w_{ij}(n)} = \frac{\partial E(n)}{\partial e_j(n)} * \frac{\partial e_j(n)}{\partial y_j(n)} * \frac{\partial y_j(n)}{\partial v_j(n)} * \frac{\partial v_j(n)}{\partial w_{ij}(n)} \tag{15}$$

Differentiating equation (11) results in:

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \tag{16}$$

Then equation (9) is differentiated with respect to $y_j(n)$, resulting in:

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{17}$$

The next step is to differentiate equation (14) concerning $v_j(n)$ and we get:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'(v_j(n)) \tag{18}$$

Differentiating equation (13) with respect to $w_{ji}(n)$ results in:

$$\frac{\partial v_j(n)}{\partial w_{ij}(n)} = y_j(n) \tag{19}$$

Applying equations (16), (17), (18), and (19) in formula (15) we obtain:

$$\frac{\partial E(n)}{\partial w_{ij}(n)} = -e_j(n) * \varphi'(v_j(n)) * y_j(n) \tag{20}$$

The correction applied to the synaptic weight ∇w will be defined by the *delta rule* [36]:

$$\nabla w_{yj}(n) = -\eta\frac{\partial E(n)}{\partial w_{ij}(n)} \tag{21}$$

Here, η represents the learning rate applied to the algorithm. Thus, the local gradient also called the sensitivity factor will be equal to equation (22):

$$\delta_j(n) = \frac{\partial E(n)}{\partial v_j(n)} \tag{22}$$

Weights are randomly initialized and then updated through back propagation. The $\delta_j(n)$ sensitivity factor is propagated backward to support the update of the synaptic weights. The error resulting from the difference between the output value of the last layer and the desired value is multiplied by φ' and the new weight w will be updated according to equation (23):

$$w_{yj(n)} = w_{yj(n-1)} + \eta\frac{\partial E(n)}{\partial w_{ij(n)}} \tag{23}$$

Thus, at each cycle or time run in the algorithm the weights w are updated to learn the function rule that takes the values from the input set to the output set. With this mathematical basis, the proposed ANN-RB network was implemented.

F. HYPERPARAMETER ADJUSTMENT

Factors that influence the error behavior, such as the number of layers, the number of neurons or the learning rate, are named hyperparameters [41], [42]. These values are not usually obtained from a rigid method, but from heuristics, which depend a lot on the experience of those who build the neural network [43]. As it is not possible to know in advance the best ANN-RB topology, it is reasonable to start from known patterns and modify them by measuring the response performance of the network to each change of its hyperparameters. A useful method in the search for efficient topologies is called Grid Search [44], [45].

The technique consists of setting up a series of experiments whose design variables have variable factors, each with discrete values and whose experimental units assume all possible combinations of these levels [46]. This step allows for a refined adjustment of the positioning sensor calibration process.

The grid search performs a training process for each combination of hyperparameters and saves the network that presents the best result (least error). In this way, the experiment was configured to scan many possible combinations of

the number of hidden layers, the number of neurons per layer, different learning rates, and activation functions (according to pre-selected values). The process can be considered brute force and is generally used in the choice of hyperparameters when there is no heuristic or reference value about the ideal set of values. The model saved the network that showed the highest accuracy at the end of the training, and the swept hyperparameters were:

1) Number of neurons per layer (50, 100, 200, 300, 400);
2) Learning rate (0.1, 0.01, 0.001, 0.0001);
3) Number of hidden layers (1, 2, 3, 4, 5).

The methodology did not define the activation function a priori, and its choice was made by the network during the training process, through the grid search. The adjustment of the network hyperparameters tested three activation functions (logistic sigmoid, hyperbolic tangent, and rectified linear).

G. TEST WITH OTHER REGRESSORS

Three other algorithms were used in this study to train the model rapidly. As the tests took a long time, it would be impractical to repeat them in various configurations using all samples. Therefore, a smaller set of data (4500 samples) was implemented in the research to solve the problem in a more objective and fast way. The idea is not to counter the classifiers but to test other settings that might solve the problem. The assessment included the following networks: MLP and XGBoost Regressor.

1) XGBoost REGRESSOR

The XGBRegressor method does not allow modeling of a particular architecture, and only a few hyperparameters were adjusted at the beginning (*GridSearchCV*), such as the learning rate and the number of estimators. The configuration hyperparameters that have been adjusted are as follows:

1) Learning rate: (0.5, 0.1, 0.05);
2) Number of estimators: (100, 200, 500, 1000);
3) Booster gbtree;
4) Maximum depth 20.

2) MLP

The MLP network was configured using the grid protocol and employed a proprietary algorithm (*GridSearchCV*) to make the selection of hyperparameters. The number of hidden layers varied between 1 and 3. The tested learning rates were 0.1, 0.01, 0.0001. The optimizer employed was Adam. The remaining hyperparameters were the same as the default provided by the developer (scikit-learn). Two metrics were used for comparison in all algorithms, i.e., the Mean Square Error and the Maximum Error.

H. HARDWARE

Backpropagation guarantees a high probability of convergence. However, the success of this convergence is proportional to the number of analysis points collected in an experiment and the computational power associated with recurrent matrix calculations. To allow the computational sufficiency necessary for the experiment, the following hardware was employed:

1) Twelve Nvidia cards of the GTX 1070 model with 1920 CUDA cores;
2) Two XEON processors with 24 CPU cores, with 64 gigs of memory and liquid cooling;
3) 1 TERA HD SSD workstation.

III. RESULTS

This section presents the results of the three test batteries with the robot performing different trajectories, using the network developed in this research in conjunction with the sensor fusion. Next, it is shown results of other neural networks of general use. The first results (before the grid search) are related to the proposed ANN-RB and were unsatisfactory, with large differences between the IMU position and the nominal position values of the robot controller, as shown in the Figure 7. The poor result shows the importance of the adjustment process and the search for the ideal hyperparameters. The initial outcomes do not suggest that the network is poorly implemented, but it does indicate the need for comprehensive training that takes into account various combinations of hyperparameters.

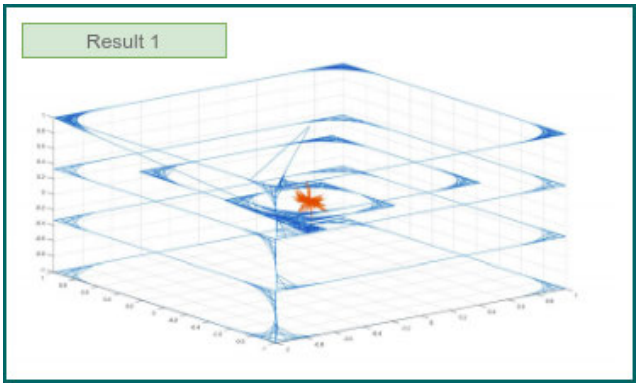

FIGURE 7. ANN-RB values (red) vs. Controller values (blue). The network did not obtain satisfactory results before the adjustment.

A. THE ANN-RB ALGORITHM

The network input is the 19 features provided by the IMU, and the output is a prediction of the TCP position provided by the algorithm. The network output is compared with the nominal value of the robot controller software. The red points in the center of the figure show what the IMU recognized as the robot's TCP position, and the blue ones are the TCP trajectories read in the robot controller. This demonstrates that the ANN-RB had not yet learned to correct the values from the errors of the IMU, and it would still be necessary to make changes to the hyperparameters of the neural network and its topology. The result plots are presented according to a subsample of the test set. The validation method is a variation of the K-Fold suitable for data sets that have a temporal relationship. The results delivered by ANN-RB were

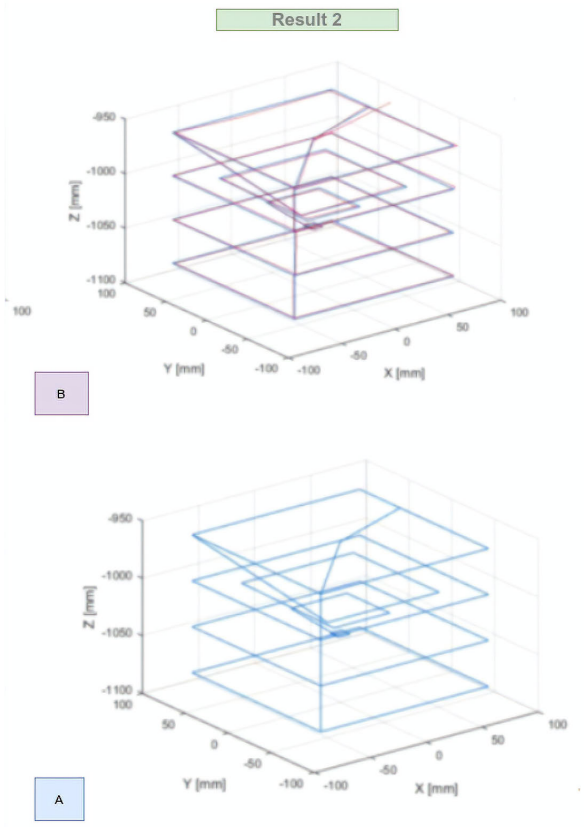

FIGURE 8. Comparison between robot controller values (A) and IMU sensors (B), considering the test set.

satisfactory (with millimeter errors) but, the training time was high due to the range of tested hyperparameters, and for this reason, the survey looked for other algorithms and means to accomplish the same task and obtain good performance with reproducible time.

After performing the grid search, the network output converged to points closer to the actual TCP position values. Figure 8 shows that the path predicted by the network is very close to the desired one. The blue line represents the robot's nominal trajectory, and the purple line describes the results obtained by the approach. The method presented a maximum error of 0.9 mm. According to the IRB-360 manual, the TCP accuracy is 0.04 mm, so the maximum error of the approach is (0.9 ± 0.04). The Mean Square Error was (0.68 ± 0.21) mm and the Root Mean Square Error was (0.82 ± 0.46) mm.

The experiment evaluated approaches on different trajectories, such as uphill, ramps, and circles at the three coordinates. Figure 9 shows circular paths. Thus, the intelligent system can perform the estimated navigation of the robot's TCP from the data provided by the fusion of sensor signals from the IMU. Figure 10 exhibits the error distribution.

The approach could predict the robot position using data obtained from the sensors with the proposed neural network, with high accuracy and submillimetric errors. The apparent disadvantage of the model is the need for a large volume of data, which resulted in a 20-day training. The training time obtained is due to two factors. First, the sample size set (6.2 million samples) is considerable. Second, the grid search analyzed several possibilities of arrangements for hyper-parameters, and this increased the time. Therefore, the methodology employed other regression algorithms adopting a smaller data set and using a smaller search protocol. Also, the results in the sequence suggest the potential of the approach, since in the worst case, it took a few minutes to perform the training.

The data points were acquired during the IMU operation, which made it possible to train the network with real data. The results obtained in the testing phase are based on these samples and, therefore, on a robot operation arrangement under real conditions. That means that it is not necessary to embed a neural network of sensors onto the robot but to use the data collected by the IMU to predict the position and correct nominal deviations.

B. THE MLP ALGORITHM

The MLP algorithm presented interesting results with a MSE of 2.81 mm and a Maximum Error of 135.2 mm. The validation process known as rolling cross-validation improved the results, and the path described in Figure 11 approached the real trajectory taken by the robot's software. The training process took approximately 5 minutes.

C. XGBoost REGRESSOR

This algorithm presented promising results, and the expected trajectory started to converge satisfactorily. The training process did not last more than a few seconds, and this is an intrinsic feature of the algorithm. The experiment tried many arrangements and the results were similar. It is possible to perceive that with the known data, the system follows the nominal trajectory of the robot, but when the values are new and contaminated with the progressive noise of the IMU, the network cannot maintain the same precision. It is possible to assume that the model has become specific or sensitive to noise amplitude. As the variation in the results was small between the different arrangements of this model, it is possible to infer that the lack of data was the limiting factor. Figure 12 shows the results obtained by XGBoost.

The table 1 shows the overall result of the experiment. Graphically, it is possible to see that all networks started to describe the real trajectory. Quantitatively, the algorithms that were trained on a limited data set showed greater errors.

TABLE 1. General results.

Algorithm	Maximum error (mm)	Mean Squared Error (mm)
XGBoost	120.3	2.34
MLP	135.2	2.81
ANN-RB	**0.9**	**0.68**

IV. DISCUSSION

The results achieved in this research demonstrate that it is feasible to perform the estimated navigation of the robot's

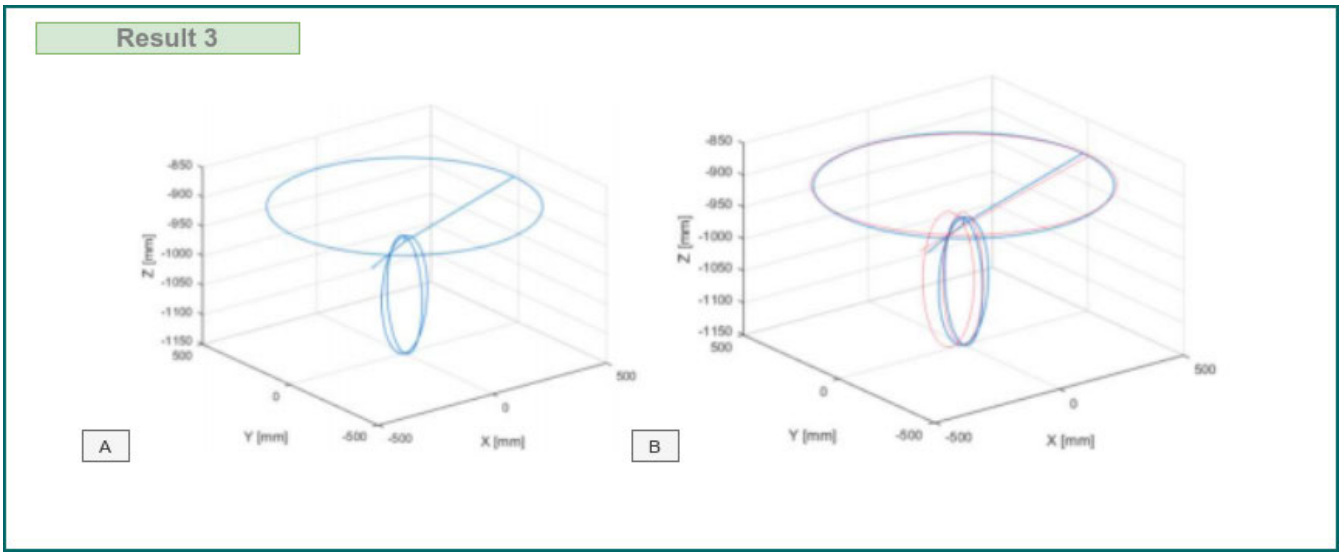

FIGURE 9. Circular path predicted by the network. In this case, the highest deviations occurred.

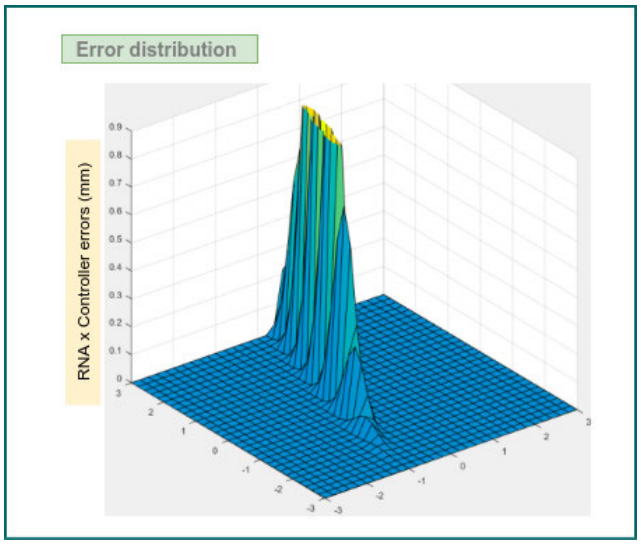

FIGURE 10. Gaussian distribution of errors.

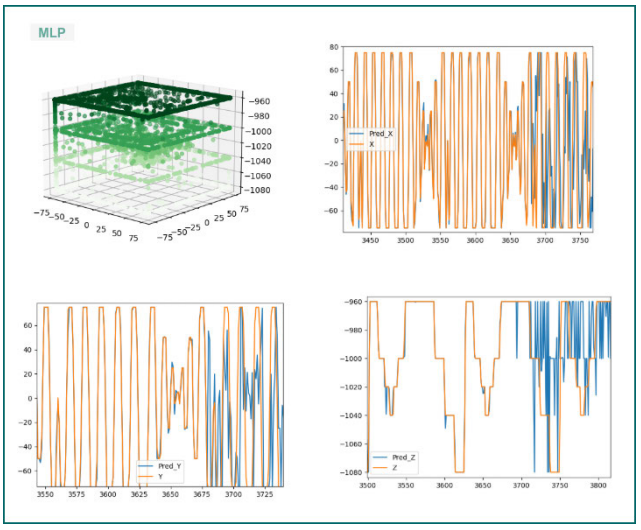

FIGURE 11. Results obtained by the MLP network.

TCP using a non-linear regression method through the fusion of micro-sensor signals. The ANN-RB network was able to accurately predict the test path, as shown in Figure 6 (with a maximum error of 0.9 mm). The study fed the regressor in serial mode, and the algorithm made the prediction instantly (online). Therefore, the methodology did not consider the forecast time (since it was insignificant).

With these results, it is viable to develop a method to perform the calibration of the robot kinematic model in a non-parametric strategy. In this research, the method performed the regression of the trajectory coordinates tracked in a guided movement. Feeding a neural network with signals from an IMU proved viable to predict the trajectory of a robot TCP in a guided movement. Even with the use of an inexpensive software solution and the acquisition of noisy measurements, it is possible to correlate nonlinear inputs in a black-box system and obtain a solution capable of measuring the robot's path with acceptable accuracy. With the regression obtained by the proposed network, the network predictions can be used with the nominal positions calculated on the controller to predict the robot position error, correcting the robot position via software in real-time whenever necessary.

Initially, as the objective was extensive training and the prospect of adjusting the network parameters the experiment provided the algorithm with a large volume of raw data. This decision was made considering that the amount of data is a limiting factor for machine learning. In the same way that this choice helped to fulfill the initial challenge and confirm the proposition, it also introduced a contradiction, i.e. a necessary training time of approximately 20 days. Despite this, it is

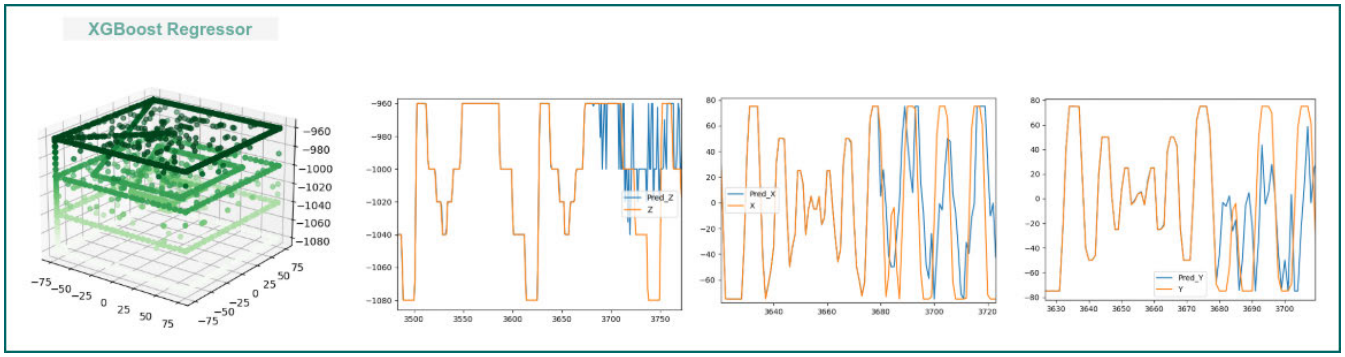

FIGURE 12. This regressor delivered the best results regarding a decreased data set, with training time in the seconds.

not reasonable to state that the model is inefficient for two reasons:

- The procedure to optimize a model and identify its ideal operational range is always expected, avoiding redundant training. In addition, one can test the network with a smaller number of samples and find a good relationship between accuracy and data volume.
- After training the network, it is possible to replicate the system without time restrictions. Supposing that in the worst case, after all the improvements, the interface needs 2 days to supply the model. At the end of that period there would be a functional and cheap product of acceptable precision.

The estimated errors were sub millimetric and acceptable for the proposal, which encourages the expansion of the research to other subjects. First, one can improve the model and look for other ways to cancel the progressive IMU noise. Second, several real applications that do not require larger accuracy can benefit from the concept, i. e. any device that needs spatial calibration over time and whose adjustment is millimetric can make use of the technique. It can also be considered that IMUs will become more and more accurate over time. The first benefit of using this method is to encourage a production line to be more efficient and to reduce undesirable downtime. In addition, a more comprehensive search for possible applications can provide insights for more general and sophisticated solutions. A complex device that can benefit from this technique would be a vehicle navigation system. In that case, the accuracy improvement may be even higher, considering that navigation methods are expensive and that it turns to be interesting to use a more accurate and less noisy IMU in conditions where the positioning accuracy may have a larger range of errors.

A. TEST WITH OTHER NETWORKS

The proposed methodology had been used with a protocol to test its results with other algorithms to investigate its feasibility and sensitivity to the data set. The idea was to verify whether it is possible to train and test other approaches faster, as some tests with the proposed one took up to 20 days. Despite worse results, the networks compared started to predict the trajectory with a shorter training time.

The algorithms used to verify the robustness of the system were able to achieve interesting preliminary results, even with a small amount of data compared to the architecture initially proposed. All regressors were able to initiate the convergence to the robot path, which means that the abstraction between the input and the output can probably be solved by these other algorithms, helping to generalize the proposed scheme. In addition, there is a drastic drop in the training time. Some observations about the networks are:

- **MLP:** The training process took a few minutes and there was little difference between different configurations of the network. The quality of the results indicate that the potential limiting factor of MLP is the reduced data set. Therefore, even before performing the research on the network, it is necessary to create a relevant and objective abstraction of the data.
- **XBoost Regressor:** This algorithm presented the most favorable results among third party networks. Accuracy is noticeably better and training time has been limited to a few seconds.

TABLE 2. Comparison table.

Method	1	2	3	4	5	ANN-RB
Maximum error(mm)	38.8	89.2	45.7	75.7	89.2	**0.9**

B. COMPARISON WITH OTHER APPROACHES

Unlike traditional (parametric) calibration methods, the noise removal process is implicit in the network, and this eliminates the need to know how this step is performed, eliminating the use of expensive IMU and sophisticated filtering mechanisms. In [47] it is presented a comparison of some parametric methods with purposes similar to this work, where the IMU is used to track the movement of a link. Although the application is different, the concept of space tracking is the same. Table 2 shows the maximum error of each approach [47].

1) The first method aims to track the position in real-time and also uses IMU signals. The article [48] introduced the technique;

2) Method 2 employs an extended Kalman filter to perform real-time tracking of human limbs by processing the signals from inertial sensors [49];
3) Technique 3 estimates linear acceleration during the walking movement and can predict the limb position. The research [50] details the algorithm;
4) Method 4 uses a recursive filtering and sensor fusion technique to do the tracking, also, it employs a camera to compensate and replace the magnetometer measurements, as described in the article [51];
5) The system 5 employs a Kalman filter in conjunction with inertial sensors to estimate the position with up to 7 degrees of freedom [52].

C. FUTURE WORK

As the data provided by the sensors was noisy and the amount of information needed to train the network was large, future work will be on software methods capable of mitigating IMU noise, making the input data smoother. Also, we want to adjust the ANN-RB architecture to consider a smaller set of hyperparameters and a smaller input series. When it is no longer possible to improve the results, the inputs will be doubled to reassess the product.

A different stage of the investigation is to develop a system that could perform an online robot calibration in realistic terms, working as a compensation model that mitigates the deviations when they are detected. For instance, supposing that the measurements obtained by the ANN-RB start to vary from the nominal values, that difference could feed the robot controller software to make corrections in the positions recursively.

V. CONCLUSION

It can be concluded from the research work that the use of IMUs and ANNs was successful to estimate the robot's TCP position and to provide measurement data for robot calibration. The approach was able to perform a nonlinear correlation between inputs and outputs in a black-box model. The IMU collected data for the ANN-RB in a sensor fusion scheme that proved to be sufficient to provide the dynamic characteristics of the guided movement. The data collected by the micro-sensors were noisy and did not accurately describe the real dynamics of the robot's TCP movement, hiding information of its signals, such as a ramp, limit switching, or change of direction. It was precisely this subjective relationship that the proposed system was able to describe, accomplishing the online calibration of the sensor. Processing time was not favorable, but this does not reflect a disadvantage, since after training the network can be easily replicated. Particularly, this study scheme used many samples to perform the tests.

The study also provided a comparison of the results with parametric approaches and this showed the capacity of the proposed method. Unlike traditional methodologies that employ mathematical means of compensation and estimation, the use of neural networks allowed to make this correlation in an automated and accurate way.

This research developed its neural network, and as the tests took too long, it was decided to examine other approaches using fewer samples. This step was carried out with different algorithms disseminated in the academic environment to test more possibilities, configurations, and networks. The results were satisfactory and showed that it is possible to generalize the model. The smaller set of data used with these networks reduced the training time, but the quality of the results decreased.

REFERENCES

[1] M. A. K. Bahrin, M. F. Othman, N. H. N. Azli, and M. F. Talib, "Industry 4.0: A review on industrial automation and robotic," *J. Teknologi*, vol. 78, nos. 6–13, Jun. 2016.
[2] M. Parigi Polverini, A. M. Zanchettin, and P. Rocco, "A constraint-based programming approach for robotic assembly skills implementation," *Robot. Comput.-Integr. Manuf.*, vol. 59, pp. 69–81, Oct. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0736584518305726
[3] L. S. Ginani and J. M. S. T. Motta, "Theoretical and practical aspects of robot calibration with experimental verification," *J. Brazilian Soc. Mech. Sci. Eng.*, vol. 33, no. 1, pp. 15–21, Mar. 2011.
[4] J. M. S. T. Motta, "Robot calibration: Modeling measurement and applications," in *Industrial Robotics: Programming, Simulation and Applications*, L. Kin, Ed. Linz, Austria: Pro Literatur Verlag, Germany / ARS, Austria, Dec. 2006.
[5] J. M. S. T. Motta, C. H. Llanos-Quintero, and R. C. Sampaio, "Inverse kinematics and model calibration optimization of a five-DOF robot for repairing the surface profiles of hydraulic turbine blades," *Int. J. Adv. Robotic Syst.*, vol. 13, no. 3, p. 114, Jun. 2016. [Online]. Available: http://journals.sagepub.com/doi/10.5772/63673
[6] R. Wang, A. Wu, X. Chen, and J. Wang, "A point and distance constraint based 6R robot calibration method through machine vision," *Robot. Comput.-Integr. Manuf.*, vol. 65, Oct. 2020, Art. no. 101959.
[7] Z. Wang, Z. Chen, Y. Wang, C. Mao, and Q. Hang, "A robot calibration method based on joint angle division and an artificial neural network," *Math. Problems Eng.*, vol. 2019, pp. 1–12, Mar. 2019.
[8] J. Peng, Y. Ding, G. Zhang, and H. Ding, "An enhanced kinematic model for calibration of robotic machining systems with parallelogram mechanisms," *Robot. Comput.-Integr. Manuf.*, vol. 59, pp. 92–103, Oct. 2019.
[9] A. Y. Elatta, L. P. Gen, F. L. Zhi, Y. Daoyuan, and L. Fei, "An overview of robot calibration," *Inf. Technol. J.*, vol. 3, no. 1, pp. 74–78, Jan. 2004.
[10] M. Slamani, A. Joubair, and I. A. Bonev, "A comparative evaluation of three industrial robots using three reference measuring techniques," *Ind. Robot, Int. J.*, vol. 42, no. 6, pp. 572–585, Oct. 2015.
[11] X. Chen, Q. Zhang, and Y. Sun, "Non-kinematic calibration of industrial robots using a rigid–flexible coupling error model and a full pose measurement method," *Robot. Comput.-Integr. Manuf.*, vol. 57, pp. 46–58, Jun. 2019.
[12] J. Sabsch, M. Hanses, S. Zug, and N. Elkmann, "Towards improving the absolute accuracy of lightweight robots by nonparametric calibration," in *Proc. 22nd IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*. Limassol, Cyprus: IEEE, Sep. 2017, pp. 1–4.
[13] W. Yang, Y. Liu, and F. Liu, "An improved relative GNSS tracking method utilizing single frequency receivers," *Sensors*, vol. 20, no. 15, p. 4073, Jul. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/15/4073
[14] S. Kolyubin, L. Paramonov, and A. Shiriaev, "Robot kinematics identification: KUKA LWR4+ redundant manipulator example," *J. Phys., Conf. Ser.*, vol. 659, Nov. 2015, Art. no. 012011.
[15] S. Marie, E. Courteille, and P. Maurine, "Elasto-geometrical modeling and calibration of robot manipulators: Application to machining and forming applications," *Mechanism Mach. Theory*, vol. 69, pp. 13–43, Nov. 2013.
[16] P.-N. Le and H.-J. Kang, "Robot manipulator calibration using a model based identification technique and a neural network with the teaching learning-based optimization," *IEEE Access*, vol. 8, pp. 105447–105454, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9108287/
[17] M. Di Cicco, L. Iocchi, and G. Grisetti, "Non-parametric calibration for depth sensors," *Robot. Auto. Syst.*, vol. 74, pp. 309–317, Dec. 2015.

[18] P. Xiao, H. Ju, Q. Li, J. Meng, and F. Chen, "A new fixed axis-invariant based calibration approach to improve absolute positioning accuracy of manipulators," *IEEE Access*, vol. 8, pp. 134224–134232, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9146124/

[19] J. Q. Xuan and S. H. Xu, "Review on kinematics calibration technology of serial robots," *Int. J. Precis. Eng. Manuf.*, vol. 15, no. 8, pp. 1759–1774, Aug. 2014.

[20] F. Farhangian and R. Landry, "Accuracy improvement of attitude determination systems using EKF-based error prediction filter and PI controller," *Sensors*, vol. 20, no. 14, p. 4055, Jul. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/14/4055

[21] V. Sonetha, P. Agarwal, S. Doshi, R. Kumar, and B. Mehta, "Microelectromechanical systems in medicine," *J. Med. Biol. Eng.*, vol. 37, no. 4, pp. 580–601, Aug. 2017.

[22] Y. Zhu and T.-H. Chang, "A review of microelectromechanical systems for nanoscale mechanical characterization," *J. Micromech. Microeng.*, vol. 25, no. 9, Sep. 2015, Art. no. 093001.

[23] J. W. Judy, "Microelectromechanical systems (MEMS): Fabrication, design and applications," *Smart Mater. Struct.*, vol. 10, no. 6, pp. 1115–1134, Dec. 2001.

[24] G. Du and P. Zhang, "IMU-based online kinematic calibration of robot manipulator," *Sci. World J.*, vol. 2013, pp. 1–10, Jan. 2013.

[25] H. Vieler, A. Lechler, and J. Grimm, "Online calibration of industrial robots using inertial sensors," in *Proc. 47th Int. Symp. Robot. (ISR)*, 2016, pp. 1–6.

[26] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation," *Found. Trends Signal Process.*, vol. 11, nos. 1–2, pp. 1–153, 2017. [Online]. Available: http://arxiv.org/abs/1704.06053

[27] Y. Xu, Y. Wang, Y. Su, and X. Zhu, "Research on the calibration method of micro inertial measurement unit for engineering application," *J. Sensors*, vol. 2016, pp. 1–11, Jan. 2016.

[28] L. Zhou, E. Fischer, C. Tunca, C. M. Brahms, C. Ersoy, U. Granacher, and B. Arnrich, "How we found our IMU: Guidelines to IMU selection and a comparison of seven IMUs for pervasive healthcare applications," *Sensors*, vol. 20, no. 15, p. 4090, Jul. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/15/4090

[29] M. A. Shah, F. Iqbal, I. A. Shah, and B. Lee, "Modal analysis of a single-structure multiaxis MEMS gyroscope," *J. Sensors*, vol. 2016, pp. 1–8, Jan. 2016.

[30] V. M. N. Passaro, A. Cuccovillo, L. Vaiani, M. De Carlo, and C. E. Campanella, "Gyroscope technology and applications: A review in the industrial perspective," *Sensors*, vol. 17, no. 10, p. 2284, Oct. 2017.

[31] S. Gonseth, F. Rudolf, C. Eichenberger, D. Durrant, and P. Airey, "Miniaturized high-performance MEMS accelerometer detector," *CEAS Space J.*, vol. 7, no. 2, pp. 263–270, Jun. 2015.

[32] S. Du, W. Sun, and Y. Gao, "MEMS IMU error mitigation using rotation modulation technique," *Sensors*, vol. 16, no. 12, p. 2017, Nov. 2016.

[33] Z. Zhou, Y. Li, J. Zhang, and C. Rizos, "Integrated navigation system for a low-cost quadrotor aerial vehicle in the presence of rotor influences," *J. Surveying Eng.*, vol. 143, no. 1, Feb. 2017, Art. no. 05016006.

[34] D. Won, J. Ahn, S. Sung, M. Heo, S.-H. Im, and Y. J. Lee, "Performance improvement of inertial navigation system by using magnetometer with vehicle dynamic constraints," *J. Sensors*, vol. 2015, pp. 1–11, Jul. 2015.

[35] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, Nov. 2018, Art. no. e00938.

[36] S. S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. New York, NY, USA: Prentice-Hall, 2009.

[37] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Proc. Artif. Intell. Mach. Learn. Multi-Domain Oper. Appl.*, T. Pham, Ed. Baltimore, MD, USA: SPIE, May 2019, p. 36.

[38] X. Wan, "Influence of feature scaling on convergence of gradient iterative algorithm," *J. Phys., Conf. Ser.*, vol. 1213, Jun. 2019, Art. no. 032021.

[39] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: https://arxiv.org/abs/1502.03167

[40] X. Ying, "An overview of overfitting and its solutions," *J. Phys., Conf. Ser.*, vol. 1168, Feb. 2019, Art. no. 022022.

[41] P. Schratz, J. Muenchow, E. Iturritxa, J. Richter, and A. Brenning, "Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data," *Ecol. Model.*, vol. 406, pp. 109–120, Aug. 2019.

[42] J. Wu, S. Chen, and X. Liu, "Efficient hyperparameter optimization through model-based reinforcement learning," *Neurocomputing*, vol. 409, pp. 381–393, Oct. 2020.

[43] W. Jia, C. Xiu-Yun, Z. Hao, X. Li-Dong, L. Hang, and D. Si-Hao, "Hyperparameter optimization for machine learning models based on Bayesian optimization," *J. Electron. Sci. Technol.*, vol. 17, no. 1, pp. 26–40, 2019.

[44] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for NAS," Dec. 2019, *arXiv:1912.06059*. [Online]. Available: http://arxiv.org/abs/1912.06059

[45] R. Ghawi and J. Pfeffer, "Efficient hyperparameter tuning with grid search for text categorization using kNN approach with BM25 similarity," *Open Comput. Sci.*, vol. 9, no. 1, pp. 160–180, Aug. 2019.

[46] P. P. Balestrassi, E. Popova, A. P. Paiva, and J. W. M. Lima, "Design of experiments on neural network's training for nonlinear time series forecasting," *Neurocomputing*, vol. 72, nos. 4–6, pp. 1160–1178, Jan. 2009.

[47] A. Filippeschi, N. Schmitz, M. Miezal, G. Bleser, E. Ruffaldi, and D. Stricker, "Survey of motion tracking methods based on inertial sensors: A focus on upper limb human motion," *Sensors*, vol. 17, no. 6, p. 1257, Jun. 2017. [Online]. Available: http://www.mdpi.com/1424-8220/17/6/1257

[48] R. Zhu and Z. Zhou, "A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 12, no. 2, pp. 295–302, Jun. 2004. [Online]. Available: http://ieeexplore.ieee.org/document/1304870/

[49] X. Yun and E. R. Bachmann, "Design, implementation, and experimental results of a quaternion-based Kalman filter for human body motion tracking," *IEEE Trans. Robot.*, vol. 22, no. 6, pp. 1216–1227, Dec. 2006. [Online]. Available: http://ieeexplore.ieee.org/document/4020379/

[50] A. D. Young, "Use of body model constraints to improve accuracy of inertial motion capture," in *Proc. Int. Conf. Body Sensor Netw.*, Jun. 2010, pp. 180–186.

[51] G. Bleser, G. Hendeby, and M. Miezal, "Using egocentric vision to achieve robust inertial body tracking under magnetic disturbances," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2011, pp. 103–109.

[52] L. Peppoloni, A. Filippeschi, E. Ruffaldi, and C. A. Avizzano, "A novel 7 degrees of freedom model for upper limb kinematic reconstruction based on wearable sensors," in *Proc. IEEE 11th Int. Symp. Intell. Syst. Informat. (SISY)*, Sep. 2013, pp. 105–110.

BENEDITO A. N. CAMPOS received the degree in mechanical engineering from the University of Brasilia (UnB), Brazil, the M.B.A. degree in economics from Fundação Getúlio Vargas - FGV, and the master's degree in mechatronics systems. He is currently pursuing the Ph.D. degree in mechatronics systems with the University of Brasilia. He is a Researcher and an Entrepreneur with 28 years of experience in the field of robotics, sensors, and artificial intelligence. His projects have been spread in bank automation, land transport, and biomedical instruments and sensors. He has experience as a teacher and speaker in technology and related areas.

JOSÉ MAURÍCIO S. T. MOTTA received the degree in mechanical engineering, in 1986, the master's degree in mechanical engineering from the University of Brasília, Brazil, in 1990, and the Ph.D. degree in robotics technology from Cranfield University, U.K., in 1999. He is currently a Full Professor with the University of Brasília (UnB). His technical-scientific contributions are focused on the fields of industrial robotics, in particular robot calibration and robotic sensing, mainly robotic vision and inertial systems. His research projects focus on the development of techniques for improving the robot positioning accuracy, robotic vision for position measurement and surface mapping, development, design and construction of dedicated robots, and experimental validation of robot offline programming. He is engaged in international cooperation agreements and joint supervision of students. He is actively engaged in academic management. He is an Ad-Hoc Consultant to international journals and boards of Research Development institutions.