Real Time Direct Kinematic Problem Computation of the 3PRS robot Using Neural Networks

Asier Zubizarreta, Mikel Larrea, Eloy Irigoyen, Itziar Cabanes, Eva Portillo

Automatic System and Control Engineering Dpt., University of the Basque Country (UPV/EHU)

Abstract

Getting a reliable calculation of the Direct Kinematic Problem (DKP) is one of the main challenges for the implementation of Real-Time (RT) controllers in Parallel Robots. In the general case, the solution of the DKP needs, among other calculated variables, the estimation of the robot pose in terms of the sensors placed on actuators. Therefore, obtaining all these variables requires the use of iterative procedures which employ high computational time.

Artificial Neural Networks have been proposed to implement the complex DKP equation mapping in the literature due to their universal approximator property. However, the proposals in this area do not consider the Real Time implementation of the ANN based solution, and no approximation error vs computational time analysis is carried out. In this work, a methodology that uses Artificial Neural Networks (ANNs) to approximate the DKP is proposed. Based on the 3PRS parallel robot, a comprehensive study is carried out in which several network configurations are proposed to approximate the DKP. Moreover, to demonstrate the effectiveness of the approach, the proposed networks are evaluated considering not only their approximation capabilities, but also their Real Time performance in comparison with the traditional iterative procedures used in robotics.

Keywords: Parallel Robots, Kinematic Problem, Artificial Neural Network

1. Introduction

Current robotic applications demand more precision, speed and load handling capabilities in order to fulfil productivity and economic goals. The classical mechanical structure used for these applications is the single kinematic chain approach of serial robots, which is composed by a single series of elements that connect a fixed base with the end effector or tool. This structure provides wide operational workspace and flexibility, although its load handling capability and dynamic performance are usually limited.

[☆]This work was supported in part by the Government of Spain under project DPI2012-32882 and UPV/EHU under grant UFI11/28

Email addresses: asier.zubizarreta@ehu.eus (Asier Zubizarreta), m.larrea@ehu.eus (Mikel Larrea), eloy.irigoyen@ehu.eus (Eloy Irigoyen), itziar.cabanes@ehu.eus (Itziar Cabanes), eva.portillo@ehu.eus (Eva Portillo)

In order to satisfy industry's requirements, in the last decade, the interest on alternative robotic structures has grown. Parallel Robots [1] have been proposed as a suitable approach for high dynamic performance tasks, i.e., high speed and acceleration, precision or high load handling tasks. Their outstanding performance in these areas is due to their multiple kinematic chains, or parallel structure, in which a moving platform, where the tool is located, is connected to a fixed one by means of several serial links. This structure provides higher stiffness than the one of serial robots, allowing load to be distributed among the limbs. Moreover, positioning errors can be compensated within the multiple limbs, and the structure can even be designed to present lower inertia by locating the actuators in the fixed base.

However, the complex, multiple loop structure of parallel robots presents some disadvantages. For instance, their workspace is much smaller than the one of serial robots, and their structure is due to present singularities in this workspace. One of the most important handicaps is related to the control of these robots, as the Direct Kinematic Problem (DKP) needs to be solved. The DKP calculates the pose of the Tool Centre Point (TCP) of the robot in terms of the motion of the actuators, and is mandatory for proper robot control. In the general case, however, this problem has no analytical solution due to the complexity of the equations, and iterative numerical procedures have to be used to estimate the position of the end effector in each control loop.

Newton-Raphson (N-R) approach is widely used for the calculation of the DKP [2]. This procedure requires an initial guess of the end effector location and an indefinite number of iterations, converging to a local solution that satisfies the complex kinematic equations of the robot. The number of iterations is highly dependant on the initial guess and its distance to the solution, and for each iteration, the Jacobian of the equation system has to be calculated and inverted. Hence, the computational cost of this approach is usually high, which represents a key issue when implementing advanced controllers for parallel robots.

Being high speed tasks one of the main areas of application of parallel robots, several works have focused on optimising the computational cost of the DKP. Three main groups can be detected among the proposed solutions. The first group of approaches is focused on reducing the complexity of the nonlinear kinematic model of parallel robots by obtaining an univariate polynomial that can be solved efficiently. Examples of application can be found for several parallel robots, such as the 4 degrees-of-freedom (dof) Schönflies platform detailed in [3], or the 3 dof reconfigurable MaPaMan [4], being the most studied one the 6 dof Gough-Stewart platform [5, 6, 7]. Although effective, these approaches are very sensitive to measurement errors in practice, as a small set of data is used to estimate the rest of the variables.

The second group is focused on introducing extra sensor data that allows an analytical solution of the DKP. For that purpose, additional sensors have to be introduced in the mechanical structure in order to measure non-actuated joint motion. This approach has demonstrated to reduce DKP computational cost significantly while maintaining high estimation accuracy [8, 9]. However, the use of extra sensors increases the cost of the robot, and adds additional complexity to its calibration.

Finally, the third group is focused on the use of approximators that simplify the highly coupled and nonlinear kinematic equations. The main hypothesis behind these approaches is that a bounded but not exact accuracy is needed when estimating the end effector pose. Hence, universal approximators can be used if the approximation error is kept within the required bounds. In this context, the use of Taylor series was proposed by Wang, *et al* [10] to calculate the DKP of the Gough-Stewart platform. Among other approaches, the use of Artificial Neural Networks (ANN) have also been proposed due to their nature as universal approximators [11]. ANNs simulate the behaviour of biological neural networks by means of a set of mathematical representations of neurons, which are grouped in interconnected layers. Each neuron combines its multiple inputs to compute its output by means of a nonlinear activation function, which allows them to learn high complex input-output data relations such as the kinematic relations of parallel robots.

Few works have focused on the use of ANN as approximators for the DKP solving in parallel robots. In these, architectures based on Radial Basis Function (RBF) [12, 13], polynomial neural networks [14] or Adaptive-network-based fuzzy inference system [15] have been proposed. However, the most popular one is the Multi Layer Perceptron (MLP) architecture, with applications in the calculation of the DKP of several parallel robots, such as the Hexa [16], the Hexapod [17, 18], the 3RRR planar robot [13] or the 3RRR spherical robot [19].

The aforementioned works demonstrate the potential of ANNs as an effective solution to the computation of the DKP in parallel robots. However, most of the works are focused on the quality of the estimation, and do not consider two important issues: the definition of the bounded error to determine the validity of the ANN configuration, and the Real Time computational cost of the ANN based approach in comparison with traditional ones [20].

This work is focused on the Real Time implementation of ANN to solve the DKP of parallel robots, providing two main contributions in this area: a) a methodology to determine the best ANN architecture is provided based on a bounded error criteria; and b) a Real Time computational cost study is carried out to determine the effectiveness of the approach and to discuss its validity for RT control applications.

For that purpose, the 3PRS parallel robot has been selected as the study case and is presented in Section 2. The DKP solving approaches based on ANN are discussed in Section 3. The ANN training procedure and approximation performance is analysed in Section 4, in which the selection criteria based on bounded accuracy error is discussed. The Real Time performance of the proposed networks is evaluated in Section 5. Finally, the most important ideas are summarised.

2. The 3PRS Parallel Robot

The 3PRS is a lower mobility parallel robot, composed by three PRS (prismatic-rotary-spherical) limbs that connect a fixed base with a moving platform where the end effector, or tool, is attached (Fig. 1). This structure provides three degrees of freedom: displacement along the global z axis, and two independent rotary motions θ_x and θ_y along the global x and y axes. However, due to the configuration of the robot, movement in the x and y axes and the θ_z angle is not null. This effect is known as parasitic motion [21].

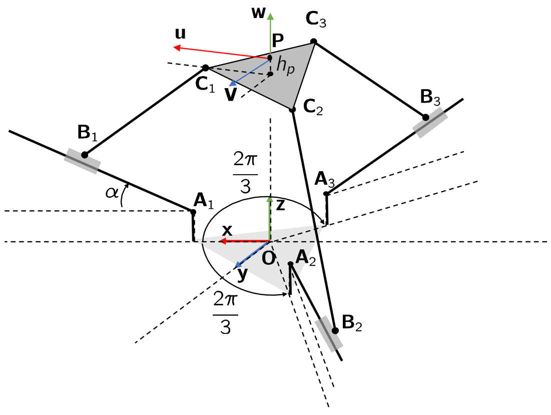


Figure 1: 3PRS Parallel Robot

There exist several applications for this robot, such as dish positioning, solar tracking for energy generation, machining or testing devices. In order to achieve the desired mobility, the robot is actuated by three linear motors $\mathbf{A_i B_i}$ that are attached to the fixed base in a symmetrical configuration. Three limbs $\mathbf{B_i C_i}$ transfer the motion of the actuators to the moving platform where the TCP $\mathbf{P}$ is located. For that purpose, one end of the limb, $\mathbf{B_i}$ is composed by a slider and a rotary joint, while the other, $\mathbf{C_i}$, presents a spherical joint. This configuration restricts the motion of the limb to the π_i plane (Fig. 2), defined by $\mathbf{A_i}$, $\mathbf{B_i}$ and $\mathbf{C_i}$. The moving platform has a regular triangular geometry, and its height is h_p.

The kinematic relations of the 3PRS parallel robot are based on the loop closure equations (Fig. 2), which relate the motion of the actuators $\mathbf{q_a}$ and the pose of the end effector located in the moving platform.

$$\mathbf{p} + \mathbf{d_i^O} - \mathbf{l_i} - \mathbf{b_i} - \mathbf{a_i} = \mathbf{0}_{3\times1}, \quad i = 1, 2, 3 \tag{1}$$

where the first two terms define the pose of the end effector, being $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ its position in the fixed frame $\mathbf{O}(x, y, z)$ and $\mathbf{d_i^O}$ the orientation vector of each vertex of the moving platform, which is calculated as,

$$\mathbf{d_i^O} = \mathbf{R}\,\mathbf{d_i} \tag{2}$$

which represents the projection of the constant vector $\mathbf{d_i} = \mathbf{PC_i}$ (defined in the moving frame) in the fixed frame,

$$\mathbf{d_i} = \begin{bmatrix} r & -r\,sin\,(2\pi\,(i-1)/3) & -h_p \end{bmatrix}^T \quad \text{i=1,2,3} \tag{3}$$

$\mathbf{d_i}$ defines the geometry of the moving platform, being $r = 0.3638$ (m) the radius of the moving platform and $h_p = 0.04$ (m) its height.

The rotation matrix $\mathbf{R}$ relates the orientation of the moving frame $P(u, v, w)$ with respect to the fixed one $O(x, y, z)$. If Roll-Pitch-Yaw $(\theta_x, \theta_y, \theta_z)$ notation is selected,

$$\mathbf{R} = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} = \begin{bmatrix} c\theta_z\, c\theta_y & c\theta_z\, s\theta_y\, s\theta_x - s\theta_z\, c\theta_x & s\theta_z\, s\theta_x + c\theta_z\, s\theta_y\, c\theta_x \\ s\theta_z\, c\theta_y & c\theta_z\, c\theta_x + s\theta_z\, s\theta_y\, s\theta_x & s\theta_z\, s\theta_y\, c\theta_x - c\theta_z s\theta_x \\ -s\theta_y & c\theta_y\, s\theta_x & c\theta_y\, c\theta_x \end{bmatrix} \tag{4}$$

where c and s stand for *cosine* and *sine* trigonometrical operations, respectively.

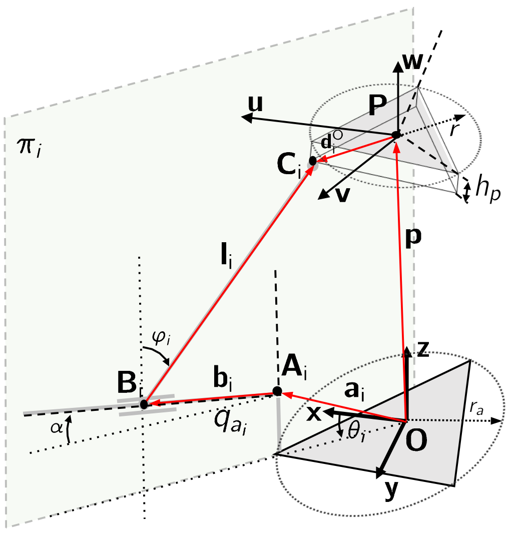


Figure 2: Kinematic loop

The three last terms of Eq. 1 are defined in terms of joint variables. $\mathbf{l_i}$ is the vector related to limb $\mathbf{B_i C_i}$, whose length $l_i = 0.9805$ (m) is constant,

$$\mathbf{l_i} = l_i \begin{bmatrix} -\cos\left(2\pi\,(i-1)/3\right)\sin\varphi_i \\ -\sin\left(2\pi\,(i-1)/3\right)\sin\varphi_i \\ \cos\varphi_i \end{bmatrix} \qquad i = 1,2,3 \tag{5}$$

$\mathbf{b_i}$ is the position vector of the slider joint $\mathbf{B_i}$, which depends on the actuated variables $q_{a_i} = |\mathbf{A_i B_i}|$ and the constant linear guide angle $\alpha = -0.7854$ (rad).

$$\mathbf{b_i} = \begin{bmatrix} q_{a_i}\ \cos\alpha\ \cos(2\pi\,(i-1)/3) \\ q_{a_i}\ \cos\alpha\ \sin(2\pi\,(i-1)/3) \\ q_{a_i}\ \sin\alpha \end{bmatrix} \qquad i = 1,2,3 \tag{6}$$

Finally, $\mathbf{a_i}$ is the constant position vector of each of the three linear guides with respect to the fixed frame, defining the fixed platform geometry,

$$\mathbf{a_i} = a_i \begin{bmatrix} r_a \cos\left(2\pi\,(i-1)/3\right) & -r_a\, sin\,(2\pi, (i-1)/3) & 0 \end{bmatrix}^T \qquad \text{i=1,2,3} \tag{7}$$

where $r_a = 0.425$ (m) is the radius of the linear guides disposition.

If a distance constraint $l_i^2 = ||\mathbf{l_i}||$ is imposed to each of the vectorial closure loop equations (Eq. 1), the relationship between the output coordinates $\mathbf{x} = \begin{bmatrix} x & y & z & \theta_x & \theta_y & \theta_z \end{bmatrix}^T$ and the input joint variables associated to the motion of the actuators $\mathbf{q_a} = \left[\begin{array}{ccc} q_{a_1} & q_{a_2} & q_{a_3} \end{array}\right]^T$ can be defined as follows,

$$\Gamma_{i+3} = q_{a_i}^2 + B_i\, q_{a_i} + C_i \;\; = \;\; 0 \quad i = 1,2,3 \tag{8}$$

where

$$\begin{array}{rcl} B_i & = & (-2\,CA_{ix}\,\cos\alpha)\,/\cos(2\pi\,(i-1)/3) \\ & & -2\,CA_{iz}\sin\alpha - 2\,CA_{iy}\,\cos\alpha\,\sin(2\pi\,(i-1)/3) \\ C_i & = & ||\mathbf{CA_i}|| + h_b^2 - l_i^2 + (2\,h_b\,CA_{ix}\sin\alpha)\,/\cos(2\pi\,(i-1)/3) \\ & & +2\,h_b\,CA_{iy}\sin\alpha\,\sin(2\pi\,(i-1)/3) - 2\,h_b\,CA_{iz}\cos\alpha \end{array}$$

being $\mathbf{CA_i} = \mathbf{p_x} + \mathbf{R}\,\mathbf{d_i} - \mathbf{a_i} = \left[\begin{array}{ccc} CA_{ix} & CA_{iy} & CA_{iz} \end{array}\right]^T$, for $i = 1,2,3$.

As stated previously, the 3PRS presents only 3 degrees of freedom, being z, θ_x and θ_y the independent output variables. In order to define the kinematic equation system of the robot, the relationship between these independent output variables and the parasite motions x, y and θ_z needs to be modelled. Hence, three constraint equations have to be introduced. As the motion of limbs $\mathbf{B_i C_i}$ is constrained to a plane π_i (Fig. 2)[21],

$$\begin{array}{rcl} \Gamma_1 & = & h_p\,w_y - r\,u_y - y = 0 \\ \Gamma_2 & = & h_p\,w_x + \frac{r}{2}\,(u_x - v_y) - x = 0 \\ \Gamma_3 & = & v_x - u_y = 0 \end{array} \tag{9}$$

where w_x, w_y, v_x, v_y, u_x and u_y depend on the RPY Euler angles $(\theta_x, \theta_y, \theta_z)$ (Eq.4).

The equation system obtained by combining Eqs. 8 and 9 defines the kinematic equations used to solve the Direct Kinematic Problem.

3. Solving the Direct Kinematic Problem

As stated previously, in robotic applications the location of the end-effector associated to the moving frame $\mathbf{P}(u,v,w)$ with respect to the fixed frame $\mathbf{O}(x,y,z)$ is essential to execute robotic tasks. As this measurement is not possible in the general case, the Direct Kinematic Problem needs to be solved, so that the output variables that define the end-effector pose $\mathbf{x}$, can be estimated in terms of the measurable actuator variables $\mathbf{q_a}$. Hence, the DKP is, in fact, a mapping problem,

$$\mathbf{x} = \mathbf{f}(\mathbf{q_a}) \tag{10}$$

where as detailed in the previous section, $\mathbf{x}$ is a vector containing the cartesian position and orientation of the platform and $\mathbf{q_a}$ is the input joint variable vector associated to the motion actuators.

However, being highly nonlinear, an analytical expression for the mapping cannot be found in the general case. The classical approach is to solve the mapping locally, by using an iterative solving procedure based on an initial guess, i.e, Newton-Raphson approach. However, it is also possible to try to define an ANN to learn the mapping between the inputs and outputs. These alternatives will be discussed next.

3.1. Classical approach: Newton-Raphson

Newton-Raphson approach is based on the use of the Jacobian $\mathbf{J}$ to iterate from an initial guess to the solution. In order to apply it to the DKP solving case, the values of the actuated joint variables $\mathbf{q_a}$ are measured and considered known, and an initial guess of all the variables in $\mathbf{x}$ (independent and parasite) is required. Then, the numerical iterative process is defined as follows,

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \mathbf{J}^{-1}\,\mathbf{\Gamma}(\mathbf{x_k}, \mathbf{q_a}) \tag{11}$$

where $\mathbf{\Gamma}(\mathbf{x_k}, \mathbf{q_a}) = 0$, for the particular case of the 3PRS robot, is the 6 equation system defined in Eqs. 8 and 9, and $\mathbf{J} = \partial\mathbf{\Gamma}/\partial x$ is the 6×6 Jacobian of the parallel robot.

The iterative procedure stops when a maximum number of iterations has been reached, or the relative error between iterations is less than a predefined value. Although it quickly converges to the solution, this approach requires inverting the Jacobian matrix of the robot, which can be time consuming.

3.2. Artificial Neural Network approach

ANNs are mathematical constructs that emulate the biological functions of neural networks. Their structure is based on artificial neurons, that emulate biological ones by the use of nonlinear activation functions. ANNs have demonstrated to be universal approximators [11], being able to obtain a bounded error in function approximation depending on the size of the hidden layer. This way, they have been used in a wide range of applications. As detailed in the introduction, in the robotics literature the Multi Layer Perceptron (MLP) structure [11] is one of the most used due to the extensive literature existing on its training and performance tuning. Hence, in this work, this architecture is selected to perform the comparative study.

It should be noted that ANNs need to be trained so that they *learn* the mapping problem. This procedure requires to define a set of examples, the Training Set, so that the weights and biases that define the parameters of the ANN can be tuned to fit the mapping problem using the Backpropagation Method. The training of an ANN is not a trivial task, and it usually is the most complex and time consuming one. The particular training procedure of the ANNs for the study case of the 3PRS parallel robot is analysed in Section 4.

Once trained, the ANN can reproduce the mapping problem after defining its inputs. Although approximation errors can arise, depending on the quality of the training, the execution of the ANN is relatively fast. The propagation stage in the neural network takes the inputs of the network and multiplies them by the weights connecting to the next layer. Once the neurons of the hidden layer have their weighted inputs added, the result passes through the activation function to propagate the signal to the next layer. All in all, the propagation stage is a set of matrix multiplications, sums and nonlinear activation functions. The computational cost of this procedure depends on the number of neurons and signals to process, and will be analysed in Section 5.

For the particular case of the 3PRS parallel robot, and the Direct Kinematic Problem mapping, different ANN topologies can be chosen. An important issue to be considered is the output configuration of the network, i.e., the number of outputs to be estimated. It seems reasonable that a complex mapping problem will require less hidden layer neurons if its outputs are estimated separately by different ANNs. Hence, based on this reasoning, three different ANN based DKP estimator approaches can be defined based on the nature of the three outputs to be obtained (z, θ_x θ_y):

- Single 3-N-3 structure, that handles the entire mapping problem in its globality (Fig.3), i.e, from the three actuator sensor positions q_{a_1},q_{a_3},q_{a_2}, the net estimates the independent output variables (z, θ_x θ_y).
- Two structures, one 3-N-1 for estimating the output position z and other 3-N-2 that provides the estimation of the output orientation (θ_x, θ_y) (Fig.4).
- Three 3-N-1 structures, one for each output (Fig.5).

The last variable to be determined is the size of the hidden layer (N). However, the lack of an established method to obtain N related to a concrete problem, requires the use of an experimental approach to determine this variable. The procedure to determine the appropriate hidden layer neuron number is analysed in the next section.

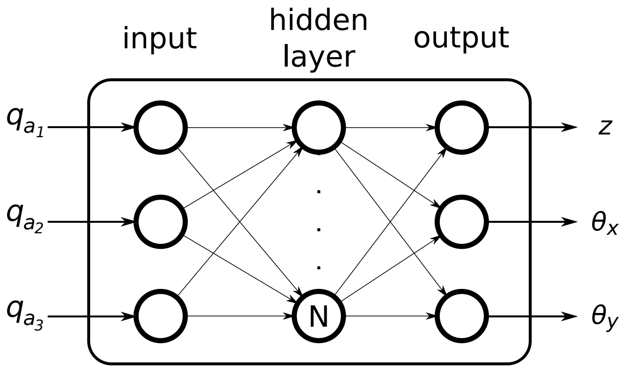


Figure 3: Single 3-N-3 net

4. ANN training methodology for DKP implementation

In order to determine the best alternative among the proposed network configurations (all coordinates $3 - N - 3$, orientation coordinates in $3 - N - 2$, and single coordinates $3 - N - 1$), an iterative batch training procedure has been designed. For this reason, for each network configuration a set of neural networks has been evaluated, whose hidden layer neuron number N varies from 5 to 90 with intervals of 5 neurons. The initialisation of the net parameters (weights and biases) has been carried out applying Nguyen-Widrow approach. Furthermore, to reduce the effect of random weight initialisation, for each value of N, 10 nets have been trained.

According to the universal approximator property as stated in [11], any type of activation function can be implemented in the designed MLP. However, due to the existence of both

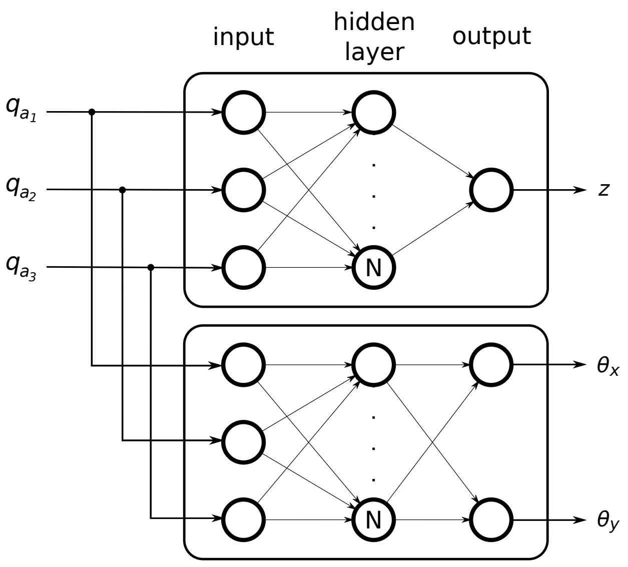


Figure 4: Position and Orientation: 3-N-1 z & 3-N-2 (θ_x, θ_y) nets

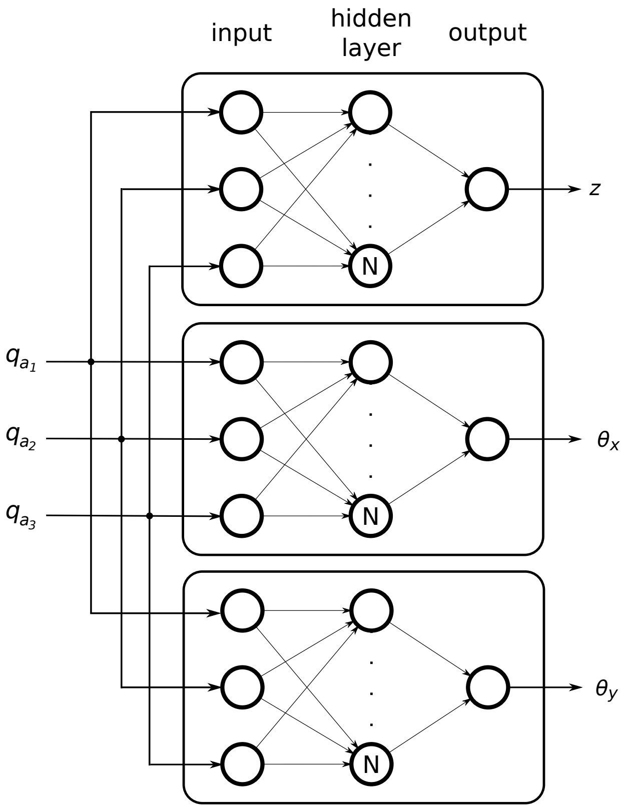


Figure 5: Three independent 3-N-1 (z,θ_x, θ_y) nets

positive and negative input-output data, the hyperbolic tangent and linear activation functions have been selected for the hidden and output layer respectively [22] for all networks

z		θ_x		θ_y	
min (m)	max (m)	min ($^\circ$)	max ($^\circ$)	min ($^\circ$)	max ($^\circ$)
0.5500	1.0500	-30	30	-30	30

Table 1: Workspace dimensions

trained.

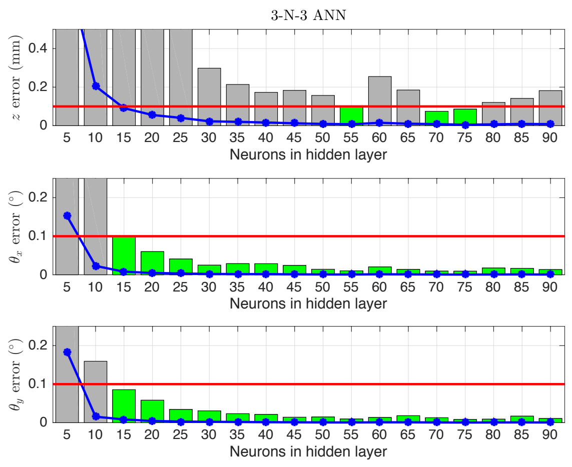
The set of training and test examples have been created based on the effective workspace of the 3PRS parallel robot. Due to the geometry of the robot, and the range of the actuators q_{a_i} $0 - 0.5$ (m), the effective workspace is contained within the boundaries defined in Table 1.

A uniform sweep of this workspace envelope has been carried out, taking 200 samples in the z coordinate range, and 50 in the θ_x and θ_y respectively. These values have been tuned experimentally based on the performance of preliminary tests, where a need to increase the samples of z coordinate has been detected. The mathematical model described in Section 2 has been used to calculate the input and output variables for each discretized point. From the set of total samples calculated, a set of 192.238 have been created for training the set of ANNs, while 2.836.219 compose the test set. It should be noted that both sets contain different examples, and have a uniform distribution in the effective workspace.

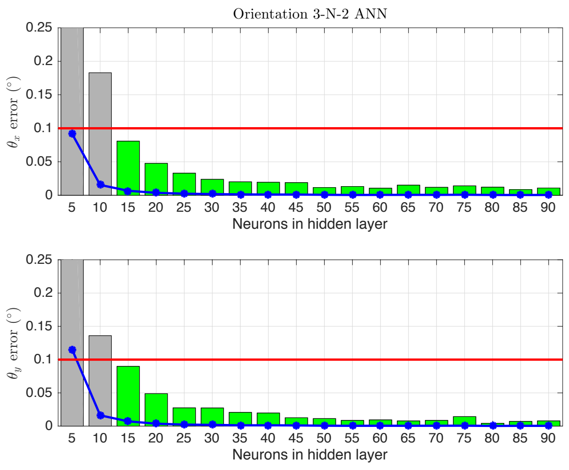
Each proposed net is trained using the previously defined sample set. For that purpose, the Levenberg-Marquard algorithm has been used with an initial adaptation rate of $\mu = 0.001$. The training has been configured to stop by gradient level criteria, establishing a 10^{-10} limit. In this process a validation phase is not performed due to a subsequent test stage, where a huge set of workspace points are estimated and evaluated, and thus, the generalisation property is preserved. Considering the different network architectures, output configurations (3-N-3, 3-N-2 and 3-N-1) and iterations, a total of 900 nets have been trained.

Once the networks have been trained, their performance has been tested using the test sample set. In order to quantify the accuracy of each network, for each output (z, θ_x, θ_y) of the network (depending on the output configuration), the *maximum absolute approximation error* is defined as the performance index. Note that the use of other indexes do not guarantee the minimum accuracy required for robot control implementation. For the particular case of the 3PRS robot, a common limit in commercial robots has been implemented, 0.1 mm for z, 0.1 $^\circ$ for θ_x and θ_y. This way, as 10 networks will be trained for each hidden layer configuration, the net with less approximation error will be selected as the representative for each network architecture (i.e, N number for hidden layer neurons), and configuration (i.e. all variables $3 - N - 3$, orientation only $3 - N - 2$ and single variable $3 - N - 1$).

Performance results for each network configuration are summarised in Figs. 6-7. The bar graph shows the maximum approximation error, while the straight blue line shows the mean approximation error for each network. The maximum admissible error limit for the approximation for this study case, i.e, 0.1 mm and 0.1 degrees, is pointed out by a red line. From the performance data shown, several conclusions can be drawn. First, due to the particular mechanical structure of the 3PRS robot and its motion range, the approximation



(a) Approximation performance of 3-N-3 nets



(b) Approximation performance of 3-N-2 nets

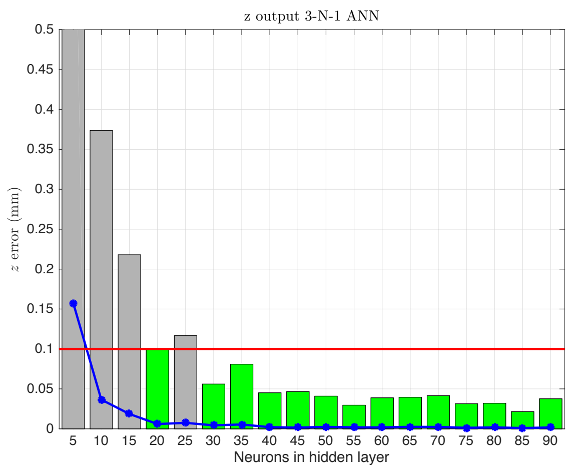
Figure 6: Approximation performance of 3-N-3 and 3-N-2 ANN architectures

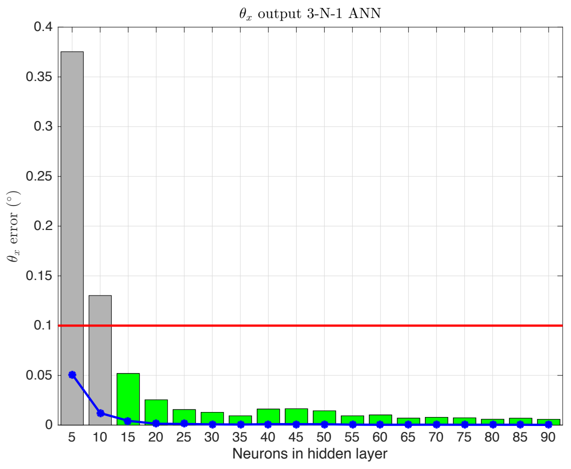
of the orientation outputs θ_x and θ_y is more accurate than the approximation of the z displacement for all network architectures. Second, simpler network configurations which estimate less number of variables present less mapping complexity, requiring less number of hidden layer neurons. These effects are both combined in 3-N-2 networks, in which an small network with 15 hidden layer neurons is able to fulfil the approximation limit constraint for both orientation outputs θ_x and θ_y, which is similar to the value required for their corresponding individual 3-N-1 networks. Finally, from these figures, the best network architectures can be deduced for each configuration. If Real-Time performance has to be considered, the nets with less number of neurons should be selected (Table 2). Note that a given configuration will only be considered as appropriate if all estimated outputs fulfil the maximum error criteria.

ANN Config.	3-N-3	3-N-2	3-N-1		
Outputs	(z, θ_x, θ_y)	(θ_x, θ_y)	z	θ_x	θ_y
N	55	15	20	15	10

Table 2: Best network architectures (hidden layer neurons)

Additional information can be deduced if the error distribution map in the workspace is analysed for each network. Figs. 8-10 represent the approximation error distribution of the best network architectures for the worst case approximations that define the performance of the network. In order to represent this mapping, the three-dimensional workspace has been divided into slices with constant z value and the distribution of errors for feasible θ_x and θ_y values has been represented using a coloured scale. As it can be seen, the maximum error approximation values can be usually found in the limits of the feasible workspace. Hence, in order to ensure proper approximation using ANNs, this fact should be considered in the training by including additional examples that emphasise training in these areas.


(a) 3-N-1 nets: z output


(b) 3-N-1 nets: θ_x output

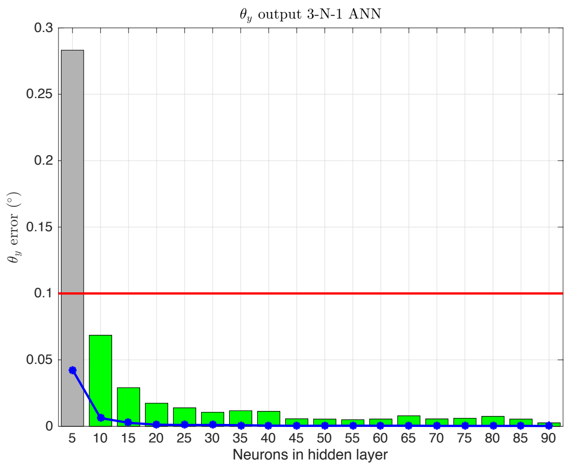
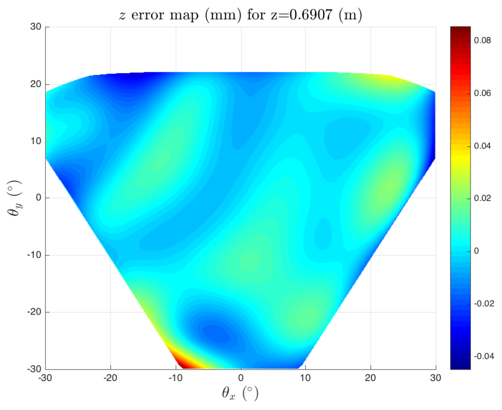

(c) 3-N-1 nets: θ_y output

Figure 7: Approximation performance of 3-N-1 ANN architectures

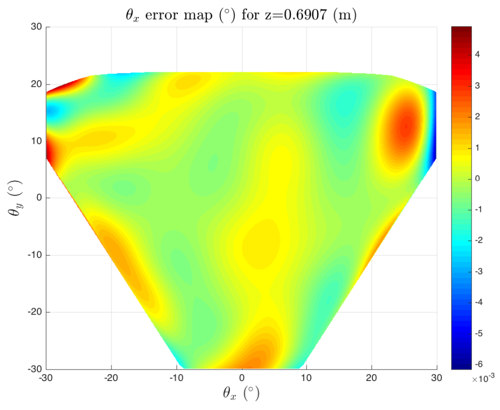
5. Real Time Performance

In this section, Real Time performance of the ANN based DKP approximators trained in the previous section is discussed. For that purpose, the time performance of each network configuration (3-N-3, 3-N-2, 3-N-1) has been evaluated in Real-Time, comparing their performance with the traditional Newton-Raphson approach.

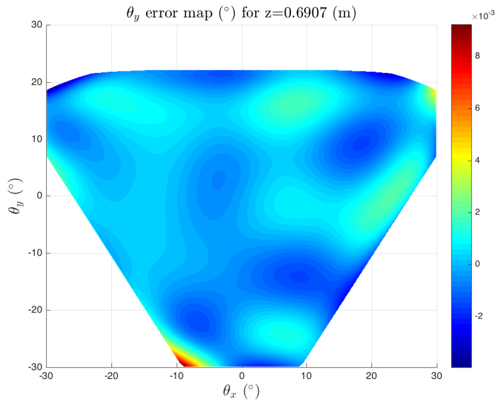
The selected evaluation platform is a 800MHz, single core, Pentium III, GEME Adlink PC104 based Industrial PC running Labview RT, Pharlap OS (Fig. 11). In order to evaluate each network performance, four time measures have been taken from the execution of each ANN, one for each required evaluation steps: the time required to normalise the inputs t_{norm}, the time required to evaluate the outputs of the hidden layer t_h, the time to evaluate the values related to the output layer t_o and, finally, the time required to denormalise these



(a) 3-55-3 net: z output

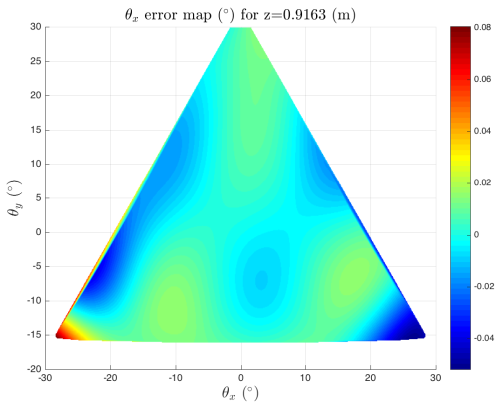


(b) 3-55-3 net: θ_x output

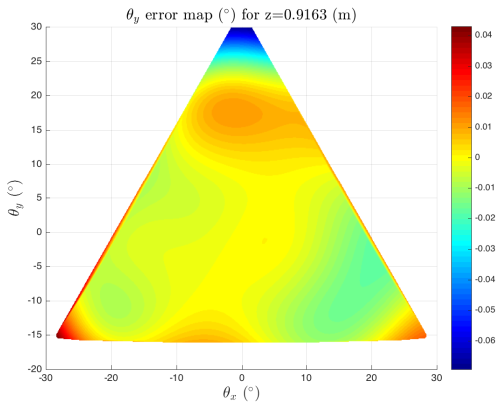


(c) 3-55-3 net: θ_y output

Figure 8: Error Distribution for the worst case approximation for the best 3-55-3 network

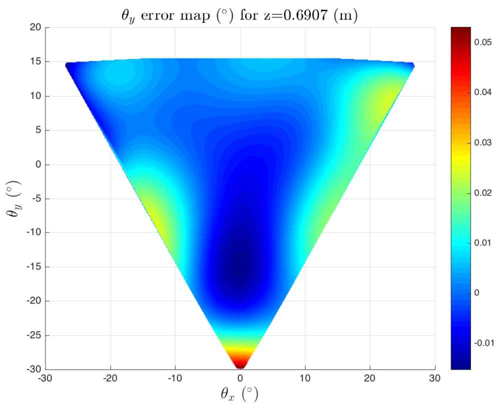


(a) 3-15-2 net: θ_x output

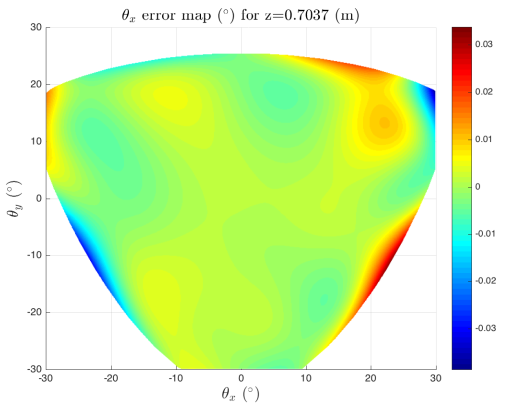


(b) 3-15-2 net: θ_y output

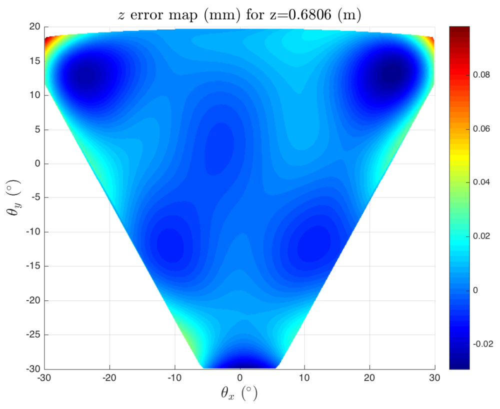
Figure 9: Error Distribution for the worst case approximation for the best orientation 3-15-2 network



(a) 3-20-1 net: z output



(b) 3-15-1 net: θ_x output



(c) 3-10-1 net: θ_y output

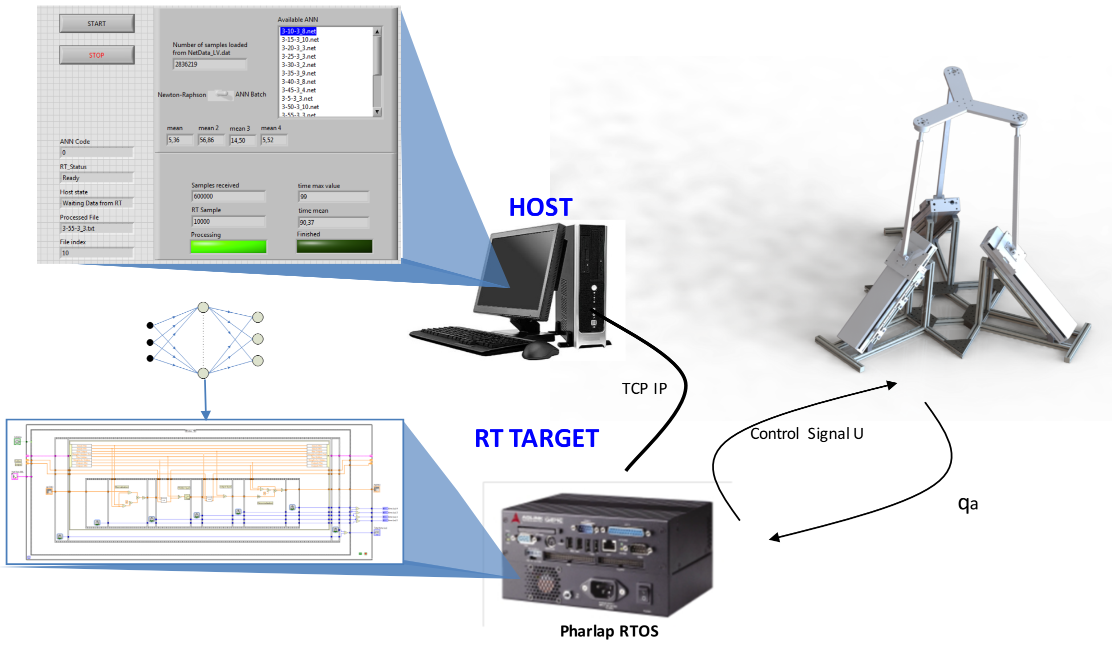
Figure 10: Error Distribution for the worst case approximation for the best single output network

outputs and get the final approximated value t_{dnorm}. The addition of these times t_{net} denotes the computational cost related to the execution of the ANN code.

This procedure has also been applied to the classical Newton-Raphson approach. To obtain equivalent performance results, the number of iterations in the NR approach has been limited by the relative maximum error defined for ANNs (0.1 mm and 0.1°). In a similar way, these limits are used to compute randomly the initial iteration point used for the N-R approach. As the execution time of a piece of code can vary slightly depending on the processor load, the time performance of each selected network, and NR approach, has been evaluated for all samples in the Test Set, and the mean execution time has been calculated.

Time performance results are summarised in Fig. 12 for each network configuration (3-N-3, 3-N-2, 3-N-1). The total bar height represents the network execution time t_{net}, and the inner stacked bars the time required to perform each step of the network evaluation.

From the comparative analysis several results can be clearly seen. Firstly, the number



Number of samples loaded from NetData_LV.dat: 2836219
Newton-Raphson / ANN Batch
Available ANN: 3-10-3_8.net, 3-15-3_10.net, 3-20-3_3.net, 3-25-3_3.net, 3-30-3_2.net, 3-35-3_9.net, 3-40-3_8.net, 3-45-3_4.net, 3-5-3_3.net, 3-50-3_10.net, 3-55-3_3.net
mean: 5,36
mean 2: 56,86
mean 3: 14,50
mean 4: 5,52
ANN Code: 0
RT_Status: Ready
Host state: Waiting Data from RT
Processed File: 3-55-3_3.txt
File index: 10
Samples received: 600000
RT Sample: 10000
Processing
time max value: 99
time mean: 90,37
Finished

Figure 11: RT System configuration

of neurons in the hidden layer is directly related to the computational cost of the approach. However, an increased number of neurons does not always guarantee better approximation as seen in previous section and an experimental study has to be carried out in order to determine the best computational cost/precision ratio.

A study of the internal net evaluation steps can be also performed. In Fig. 12, it can be seen that for a given number of neurons in the hidden layer, there are only slight time performance differences between network configurations (2-4 μs), as only the output layer size varies. Hence, t_{norm} and t_h times are identical for a given number of hidden neurons, while t_o and t_{dnorm} increase with the number of outputs. However, due to the relative small size of the networks, the effect of the output vector dimension can be neglected.

The previous analysis considers each network time performance independently. However, in order to implement an ANN based approximator, more than one ANN could be needed. Table 3 summarises the total time required for the best proposed ANN approximator approaches (single network, orientation and distance networks and three individual networks) and the traditional Newton-Raphson approach. From these results it can be clearly seen that the ANN-based approaches efficiency is at least 5 times higher than the traditional Newton-Raphson one, which makes these approaches an interesting solution for the DKP approximation. Moreover, although the training and network architecture is more complex, with 55 neurons in the hidden layer for the best case, a single 3-55-3 network is the most cost effective approach, as considering different ANNs for each output, or dividing orientation and distance outputs require more than one ANN.

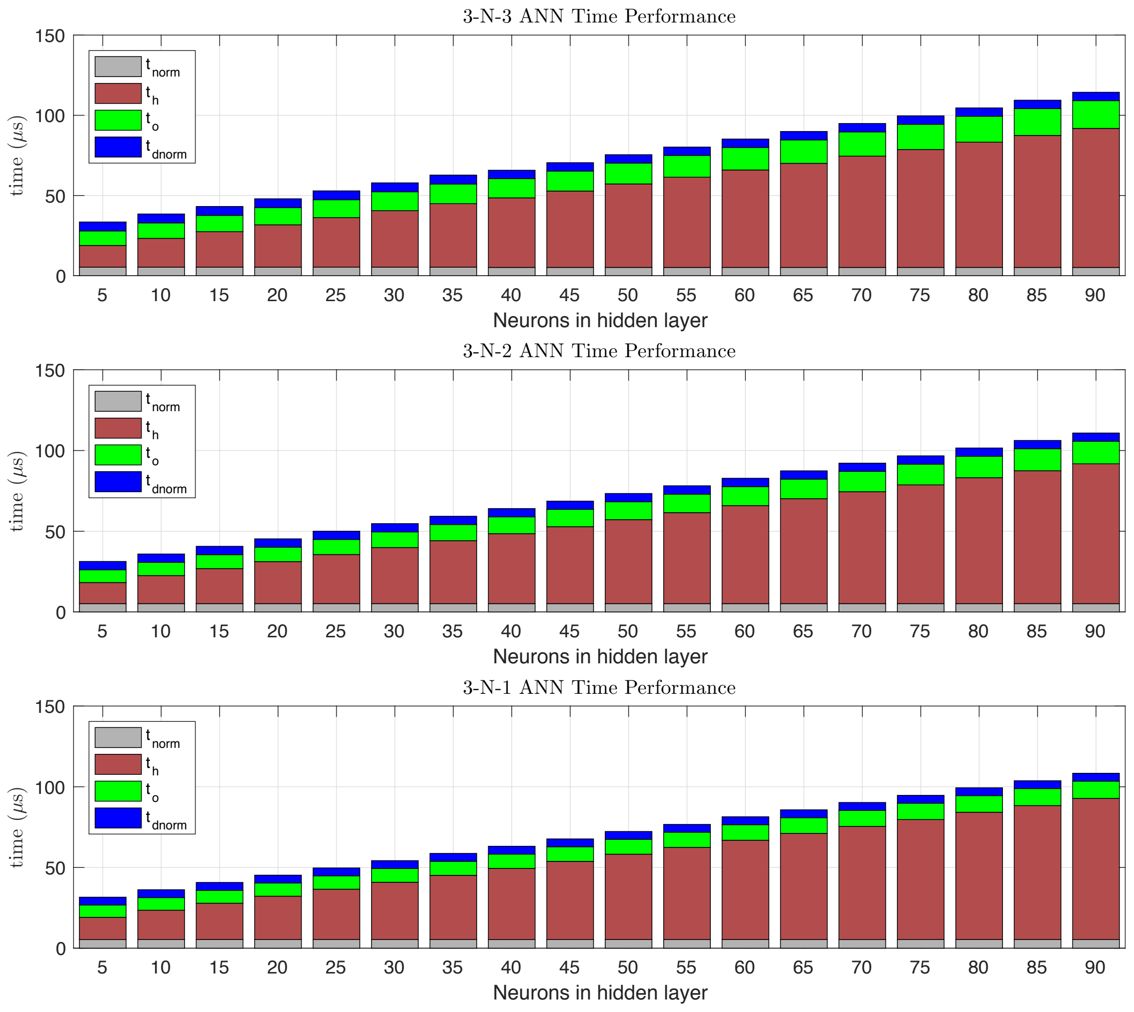


Figure 12: ANN Time Performance vs hidden neuron number

Approach	Single	Position+Orientation		Independent			N-R
Configuration	3-55-3	3-20-1 z	3-15-2	3-20-1 z	3-15-1 θ_x	3-10-1 θ_y	
Time (μs)	80.19	45.24	40.65	45.24	39.77	36.18	675.71
		85.89		121.19			

Table 3: Total Performance Times for each DKP solving approach

6. Conclusions

An efficient calculation of the Direct Kinematic Problem in Parallel Robots is critical for designing Real Time controllers. This effectiveness is related to obtaining precise robot pose values below a maximum error limit, at each sample time of execution.

Although the classical approach to solve the DKP is the Newton-Raphson algorithm, paradigms such as ANN have also been proposed. ANNs provide advantages over the NR approach, as their computational cost is lower. However, they provide an approximation to the solution, and a proper training procedure has to be defined to achieve an appropriate accuracy.

In this work, a comprehensive study of Multilayer Perceptron Artificial Neural Network accuracy and time performance in DKP solving for a 3PRS parallel robot has been presented. Taking into account the mapping problem presented by this robot, several ANN configurations have been proposed: a single 3-N-3 network, two networks (position and orientation), and a single network for each output. Each configuration has been tested with a number of hidden layer neurons in the range [5-90].

For the training process, a set of examples for ANN tunning (192.238) and ANN testing (2.836.219) have been picked from the 3PRS effective workspace, being different examples in both sets. Furthermore, in order to avoid the dependency of ANN parameters initialisation, each of those structures were trained 10 times, selecting the best iteration as representative. The maximum approximation error has been considered as the performance index in order to define the best networks.

To extract conclusions about the goodness of ANN in this problem, the networks have been evaluated in accuracy and in Real-Time, comparing the results with the classical N-R method. This way, the best structures for each configuration have been: 3-20-1 for z position, 3-15-1 for θ_x orientation, and 3-10-1 for θ_y orientation; 3-15-2 for orientation and 3-20-1 z for position; 3-55-3 for position and orientation coordinates. It is necessary to note that the worst pose estimation results have been obtained close to the workspace limits, where points with singularities emerge. For further works, a previous study could enhance the distribution of pose coordinates chosen as examples for the training process.

Other relevant aspect of this work is the analysis made in computational cost, during execution in a Real time platform. In order to evaluate the performance and efficiency of both methods, N-R and ANN, a deep analysis in time consuming has been made in a Industrial PC. Considering that the introduced code in such platform will consume different amounts of time in each phase, four steps have been evaluated separately. The most relevant value has been located into the time (t_h) required to evaluate the outputs of the hidden layer. Obviously, the entire sum of the execution time increases as the number N of hidden layer neurons grows up. As it has been presented in section 5, the structure 3-55-3 has reached the less time consuming rate, being into 80.19 μs, obtaining a value below a level of magnitude given by N-R method. These experiments validate the usage of the ANNs to solve the DKP both considering computational cost and error performance.

References

[1] J. P. Merlet, Parallel Robots, Springer, 2006.

[2] B. Siciliano, O. Khatib (Eds.), Handbook of Robotics, Springer, 2008.

[3] O. Altuzarra, A. Zubizarreta, I. Cabanes, C. Pinto, Dynamics of a four degrees-of-freedom parallel manipulator with parallelogram joints, Mechatronics 19 (2009) 12691279.

[4] R. Srivatsan, S. Bandyopadhyay, On the position kinematic analysis of mapaman: a reconfigurable three-degrees-of-freedom spatial parallel manipulator, Mechanism and Machine Theory 62 (2013) 150–165.

[5] C. Innocenti, Forward kinematics in polynomial form of the general stewart platform, Journal of Mechanical Design 123 (2) (2001) 254–260.

[6] T.-Y. Lee, J.-K. Shim, Algebraic elimination-based real-time forward kinematics of the 6-6 stewart platform with planar base and platform, Proceedings of the 2001 IEEE International Conference on Robotics and Automation 2 (2001) 1301–1306.

[7] X. Huang, Q. Liao, S. Wei, Closed-form forward kinematics for a symmetrical 6-6 stewart platform using algebraic elimination, Mechanism and Machine Theory 45 (2) (2012) 327–334.

[8] H. Saafi, M. Laribi, S. Zeghloul, Forward kinematic model improvement of a spherical parallel manipulator using an extra sensor, Mechanism and Machine Theory 91 (2015) 102–119.

[9] A. Mahmoodia, A. Sayadib, M. B. Menhajc, Solution of forward kinematics in stewart platform using six rotary sensors on joints of three legs, Advanced Robotics 28 (1) (2014) 27–37.

[10] Y. Wang, A direct numerical solution to forward kinematics of general stewart-gough platforms, Robotica 25 (2007) 121–128.

[11] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural networks 2 (5) (1989) 359–366.

[12] D. Zhang, J. Lei, Kinematic analysis of a novel 3-dof actuation redundant parallel manipulator using artificial intelligence approach, Robotics and Computer Integrated Manufacturing 27 (1) (2011) 157–163.

[13] H. Bidokhti, J. Enferadi, Direct kinematics solution of 3-rrr robot by using two different artificial neural networks, in: Proceedings of the 3rd RSI International Conference on Robotics and Mechatronics, 2015, pp. 606–611.

[14] R. Boudreau, S. Darenfed, C. Gosselin, On the computation of the direct kinematics of parallel manipulators using polynomial networks, IEEE Transactions on System, Man and Cybernetics 28 (2) (1998) 213–220.

[15] H. Sadjadian, H. D. Taghirad, Comparison of different methods for computing the forward kinematics of a redundant parallel manipulator, Journal of Intelligent and robotic systems 44 (3) (2006) 225–246.

[16] M. Dehghani, M. Ahmadi, A. Khayatian, M. Eghtesad, M. Farid, Neural network solution for forward kinematics problem of hexa parallel robot, Proceedings of the 2008 American Control Conference (2008) 4214–4219.

[17] C. seng Yee, K. bin Lim, Forward kinematics solution of stewart platform using neural networks, Neurocomputing 16 (4) (1997) 333–349.

[18] L. H. Sang, M.-C. Han, The estimation for forward kinematic solution of stewart platform using the neural network, Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems 1 (1999) 501–506.

[19] G. Liu, Y. Wang, Y. Zhang, Z. Xie, Real-time solution of the forward kinematics for a parallel haptic device using a numerical approach based on neural networks, Journal of Mechanical Science and Technology 29 (6) (2015) 2487 2499.

[20] J. Misra, I. Saha, Artificial neural networks in hardware: A survey of two decades of progress, Neurocomputing 74 (2010) 239–255.

[21] J. A. Carretero, R. P. Podhorodeski, M. A. Nahon, C. Gosselin, Kinematic analysis and optimization of a new three degree-of-freedom spatial parallel manipulator, Transactions-American Society of Mechanical Engineers Journal of Mechanical Design 122 (1) (2000) 17–24.

[22] M. Hagan, M. Menhaj, Training feed-forward networks with the marquardt algorithm, IEEE Transactions on Neural Networks 5 (1994) 989–993.