Direct Kinematics of Articulated Parallel Manipulators Using Neural Networks

A. De Luca, R. Mattone, M. Sciandrone

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Eudossiana 18, 00184 Roma, Italy

Abstract

In contrast to the case of serial chain manipulators, closed-form solutions for the direct kinematics of parallel manipulators are generally not available. Although algorithms exist in order to find closed-form solutions for particular manipulator structures, only iterative methods can be used in general. The computation times involved with numerical methods can be unacceptable in real time applications such as control. The capability of multilayer neural networks to approximate functions with any desired accuracy suggests their application to this problem. Since exact knowledge of the inverse kinematics is available in this case, it is possible to efficiently train the network by generating a sufficiently rich set of training points. This approach has been applied to the Hughes SmartEE, a 6D parallel manipulator with all rotary joints. Six of them are actuated, while the other ones are passive and assume the unique position which complies with the closed chain structure of the mechanism. The network generates these unknown values which are used to produce the desired unique solution to the direct kinematics problem. A novel optimization-based training algorithm is introduced, that overcomes the problem of local minima through the use of a regularization procedure. Numerical results are reported in order to evaluate the performance of the obtained neural network in the manipulator workspace. Special care is taken in handling passive joints values generated by the neural network, so to comply with mechanical feasibility.

1 Introduction

Neural Networks (NN) are being used in robotics and in control systems for many applications where traditional approaches are not satisfactory. In particular, NN have found large use in dynamic system identification [1,2] and in solving robot motion through learning [3–5]. As concerns robot kinematics, neural networks have been applied for solving the standard inverse kinematic problem [6]. A more interesting use is whenever a closed-form solution is not available. From this point of view, it is paradigmatic the inverse kinematics problem for serial redundant robots [7]. On the other hand, there is an increasing interest in parallel manipulators due to fast and accurate motion capabilities. Parallel manipulators typically consist of a moving platform held up by 'legs' connected to a fixed base. Each leg can be viewed as an open kinematic chain with rotative or prismatic joints, but the overall mechanism obviously contains closed loops. In general, only some of the joints are *active*, i.e. arbitrary values can be imposed though an actuator device, while the rest are *passive* ones in order to guarantee the needed mobility of the structure. When deriving the direct kinematics of such manipulators, the geometric closure constraints yield nonlinear equations which in general cannot be solved in closed form in terms of passive joints values.

Some methods have been proposed to find a closed form solution for the direct kinematics of special parallel manipulator structure. For example, Innocenti and Parenti-Castelli [8] found the solution for a typical Stewart platform, in terms of the roots of a 16th order polynomial. This method, that could be extended also to other classes of parallel manipulators, seems to be computationally intensive for real time control applications, where the moving platform pose has to be known, at every sampling time, based on joint measures. An alternative interesting method has been proposed in [9,10]: if possible, a solution in triangular form for the *dependent* joint variables of any independent loop of the mechanism, is found through the application of *projection operators* to the closure equations.

In the general case only iterative methods are available for solving the geometric closure equations. A short description of some iterative methods and of their application to this problem can be found in [11]. The solution is typically based on *gradient* and *Newton* schemes, suitably reduced and modified on the

basis of the problem peculiarities, so to significantly reduce computation time. The main disadvantage is that these schemes need to start from a good estimation of the real pose.

The idea of using neural networks in this context arises from the necessity of having a direct, computationally cheap solution, even when a closed form is not available and an approximating scheme becomes necessary. Several theoretical results are available, [12–16] showing that NN of different form and complexity can arbitrarily approximate a continuous vector function. The opportunity of using a NN as an approximator of the direct kinematic mapping of parallel manipulators, is justified also by the availability of their inverse kinematic solution. This allows to train the network generating an arbitrary number of admissible sets of joint variables. In every admissible set two subsets of *independent* and *dependent* variables can be individuated, which are respectively the *'active'* and *'passive'* joints of the structure. The network has been trained with the error between the computed passive joints and the actual values as obtained from the inverse kinematics: as a result a NN will have as input and output respectively the active and passive joints values.

It should be stressed that the approximation of the solution to the closure equations is a critical issue. Even small errors lead in fact to non-admissible set of passive joint angles and thus to undefined physical poses for the mechanism. On the other hand, in order to evaluate the practical performance of the trained network, a *'computed pose'* must be defined even for those set of joint angles which do not exactly satisfy the closed-loop constraints. These problems are not encountered in solving inverse kinematics of serial robots: actually, in this case, a set of joint angles, even if only approximating the desired solution, is anyway admissible.

We present here a neural network solution to the direct kinematics of the SmartEE [17], a 6-d.o.f. parallel platform. This robot is available in our Robotic Laboratory at DIS and used in a cell for experiments on robot cooperation.

The paper is organized as follows. The SmartEE architecture and kinematics are described first. The neural network structure and its training strategy are then illustrated. In particular, we present a novel weight updating algorithm and the choice of the training set. Numerical results are shown.

2 The parallel manipulator SmartEE

SmartEE is a 6-dof parallel manipulator originally developed at Hughes with three equivalent and symmetric legs used for the arbitrary positioning and orienting of a top plate. Each leg consists of two links and three joints. A 2-dof joint with orthogonal axes located at the leg base allows a pitch and roll motion of the first link. These two degrees of freedom are actuated by two DC motors mounted on the fixed base through a differential drive mechanism. A second 1-dof joint (elbow joint) results in a pitch motion for the second link. This joint is not actuated, and assumes the unique admissible angular position corresponding to a given set of values for the active joints. Finally, a passive spherical joint connects the leg and the top plate. There is a total of 6 active single dof joints in the robot, while the remaining passive joints are used to accommodate for the position and orientation closure constraints. In Fig. 1 a a CAD image of SmartEE is shown, created by *RobLan*, a programming language for graphic simulation of robotic systems [18]. Fig. 2 gives a schematic representation of the mechanism for its kinematic description.

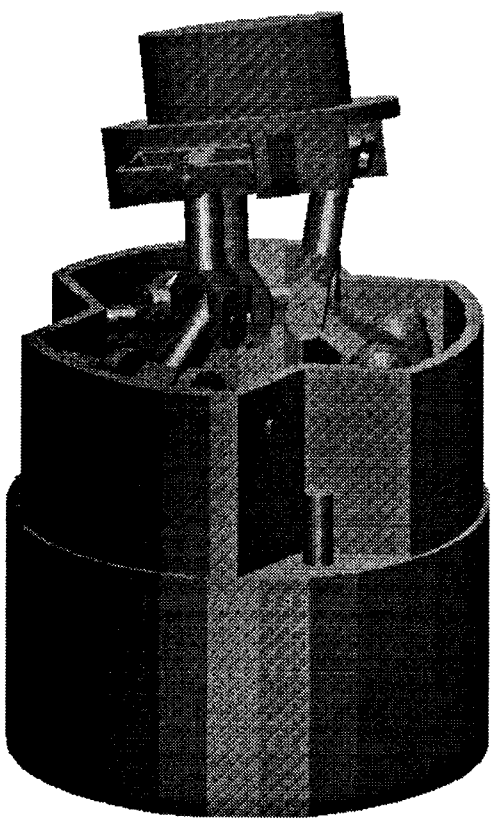

Fig. 1 – A CAD image of SmartEE

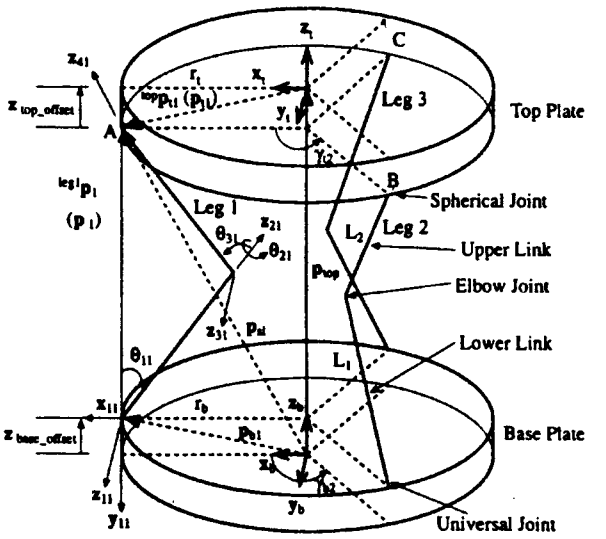

Fig. 2 – Schematic representation of SmartEE

The inverse kinematics problem of this parallel manipulator can be stated as follows: given the *pose* (position of the center of the plate and orientation of a frame attached to it) of the top plate, compute the joint angle values for each leg. The closed-form solution can be found in two steps: first, compute the tip position of each leg in a local frame; then, solve for the joint angles of each leg using the standard inverse kinematics of a 3-dof serial manipulator. The solution of inverse kinematics is unique for the SmartEE, taking into account its joint range limitation.

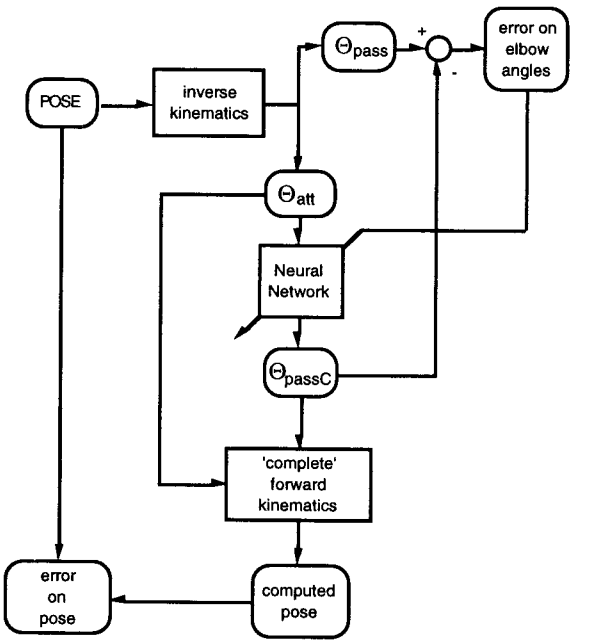

Fig. 3 – Logical scheme of training and validation

The direct kinematics problem consists in computing the top plate pose starting only from the knowledge of the two active joint angles of each leg. The solution proceeds through the evaluation of the passive elbow joint angle for each leg. Actually, if these angles were known, then we could compute directly the tip position of each leg and from these the position of the center plate and its absolute orientation: we call this the *complete* direct kinematics of the parallel manipulator, which is parameterized in terms of the passive joint angles. The actual values of the elbow angles can be found by imposing the following three closure equations (see also Fig. 2):

$$\begin{aligned} f_1 &= \|\mathbf{p}_{s1} - \mathbf{p}_{s2}\|^2 - \|\overline{AB}\|^2 = 0, \\ f_2 &= \|\mathbf{p}_{s2} - \mathbf{p}_{s3}\|^2 - \|\overline{BC}\|^2 = 0, \\ f_3 &= \|\mathbf{p}_{s3} - \mathbf{p}_{s1}\|^2 - \|\overline{CA}\|^2 = 0, \end{aligned} \tag{1}$$

where $p_{sj} = p_{sj}(\theta_{1j}, \theta_{2j}, \theta_{3j})$ is the position vector of the jth leg tip. These are three nonlinear scalar equations in the three unknowns of vector $\theta_{pas} = (\theta_{31}, \theta_{32}, \theta_{33})$, representing the passive joints of the structure, and cannot be solved in a closed form (see the Appendix for a complete expression of $\mathbf{p}_{sj}$). Let $\theta_{act} = (\theta_{11}, \theta_{12}, \theta_{13}, \theta_{21}, \theta_{22}, \theta_{23})$ be the actuated dof of the structure. It follows that the analytical expression for function $\mathbf{G}$ defined by

$$\theta_{pas} = \mathbf{G}(\theta_{act}) \tag{2}$$

is not known. Function $\mathbf{G}$ is what we would like to approximate, at least in a region of interest, by a neural network, having available an arbitrary number of admissible couples $(\theta_{act}, \theta_{pas})$. The logical NN-based computation scheme is shown in Fig. 3.

3 Feedforward multilayer neural network

A Feedforward Multilayer Neural Network (FMNN) is a neural network with neurons arranged in adjacent layers connected through weighted unidirectional links. It is known [12–14] that a FMNN with at least one hidden layer, whose outputs are bounded and monotone increasing functions, is capable of approximating any continuous mapping between finite dimensional spaces up to any desired degree of accuracy. In our case we have approximated the unknown function

$$\mathbf{G} : I\!R^6 \rightarrow I\!R^3 \tag{3}$$

using a FMNN with just one hidden layer with M neurons and hyperbolic tangent as sigmoidal output function of the neurons.

3.1 An optimization-based training algorithm

The problem of training a FMNN of a given topology is equivalent to minimizing an objective function $E(w)$, where $w \in \mathbb{R}^N$ is the vector of unknown weights [19]. In our case, $N = (6+1) \times M + (M+1) \times 3$. Two Back Propagation (BP) methods are used to solve this problem: batch and online. BP-batch is a gradient-type method, but is not based on a sound theoretical basis, is very inefficient and unreliable. BP-online is not related to any standard optimization method,but has proven convergence properties [20]. Moreover it seems to be suitable for problems with training sets characterized by redundant information. Although BP-online is the most widely used technique in applications, there are no theoretical results about its convergence rate. From this point of view, classical optimization methods, such as Conjugate Gradient or Quasi-Newton methods, have better performance even for very large dimensional problems [21,22]. Training convergence problems are given by the presence of local minima in the objective function $E(w)$. BP-online introduces some sort of randomness that may help escaping from local minima. In order to handle the local minima phenomenon in an optimization scheme, it is convenient to take advantage of the particular structure of the objective function $E(w)$ which is a composed function of sigmoidal terms. Points sufficiently far from the origin of the weight space, belong to the so-called *saturation* zone, and are likely to provide stationary points of $E(w)$. This suggests the choice of the initial guess of weights in a neighborhood of the origin [23].

We propose an alternative approach based on the minimization of a function $\tilde{E}(w)$, called *current objective function*

$$\tilde{E}(w) = E(w) + p||w||_2^2 \qquad (4)$$

obtained summing to $E(w)$ a *perturbation term* $||w||_2^2$ that penalizes the convergence towards points in the saturation zone. The parameter p assesses the influence of the perturbation term. If the global minimum point of $E(w)$ belongs to the above zone, then the influence of the perturbation term must be diminished in an iterative fashion. Consequently, after a sufficient number of iterations, $\tilde{E}(w)$ will practically coincide with the true objective function $E(w)$. The choice of the initial point is not anymore constrained, and convergence towards stationary points does not preclude the possibility of continuing the search of a global minimum point of E(w). Based on these considerations, we have defined an algorithm whose conceptual scheme is the following, where ε is the estimated minimum value of $E(w)$:

Data. w_0, $p > 0$, $\tau \in (0,1)$, $\varepsilon > 0$, $\delta > 0$, $\sigma > 0$.

Step 1. Starting with initial point w_0, minimize

$$\tilde{E}(w) = E(w) + p||w||_2^2$$

and let w^* be the obtained point

Step 2. If $E(w^*) \leq \varepsilon$, then "successful training" and stop

Step 3. If $\|w_0 - w^*\| \leq \delta$ and $\left|\frac{E(w_0)-E(w^*)}{E(w^*)}\right| \leq \sigma$, then "failed training" and stop

Step 4. If $E(w^*) < E(w_0)$ then $w_0 := w^*$

Step 5. $p := \tau p$ and go to Step 1.

Note that Step 3 is a normal stopping criterion, while Step 4 allows us to restart with the point corresponding to the smallest obtained value of $E(w)$. In this conceptual scheme, the choice of the minimizing method in Step 1 is irrelevant. Since realistic NN applications often involve adjustment of several thousand of weights and one should adopt only optimization methods suitable for large-scale problems. In particular, we have used here a pre-conditioned Conjugated Gradient Method [24].

3.2 Training Set Generation

An admissible set of joint angles for the SmartEE platform can be obtained from any pose in the workspace by means of the inverse kinematic mapping. In building the training set, we have started by choosing uniformly distributed poses in the region of interest. In order to induce in the training set the *triangular symmetry* which is proper of the mechanism, from each of the obtained sets of admissible joint angles, five other were generated by all values permutations with respect to the three legs. Furthermore, only a subset is explored in the joint space, corresponding to a proper 6-dimensional'hypercube' centered in the cartesian workspace. The linear dimensions are 2.7 cm, 2 cm and 1.5 cm respectively in the $x-$, $y-$ and $z-$direction; the angular dimensions are $\pm 10°$ of the $ZYZ-$Euler angles. This dextrous workspace is the one of interest in motion control applications, since it can be assumed that the top plate is kept away from workspace borders where kinematic singularities exist. As a result, a training set of cardinality 10^3 has been generated and a FMNN with M = 40 neurons in the hidden layer has been used. This network size, i.e. the number of hidden neurons has been chosen experimentally so to avoid overfitting or underfitting of the training set, in dependence of

the chosen parameter ε of the training algorithm.

4 Numerical Results

Neural network performance must be evaluated in terms of its generalization capability. For this purpose, we have generated a validation set whose points correspond to 288 cartesian poses in the dextrous workspace. Fig.4 shows the 2-norm of the error on the elbow angles

$$\sqrt{\sum_{j=1}^{3}(\theta_{3j} - \theta_{3jC})^2} \qquad (5)$$

where the peak value is about $3 \cdot 10^{-3}$ rads and C is for 'computed'.

On the other hand, the practical validity of the approach has to be evaluated in terms of the resulting errors on the cartesian pose. To this purpose we define first the *computed position* and computed orientation of the top plate. From the network outputs (the elbow angles), together with the network inputs (the active joints angles) we compute the tip position $\mathbf{p}_{jC}$ of the three legs. Even if errors are small, resulting tip positions will not satisfy the closure equations (1). Being the 9-ple $(\theta_{act}, \theta_{pasC})$ not admissible, the top plate pose could not be defined. Nevertheless, as $\|f_j\|$ is 'small' for every j, the *computed position* $\overline{\mathbf{p}_C}$ of the top plate center can be defined as the geometrical barycentre of the three $\mathbf{p}_{jC}$. The *'computed orientation'* of the top plate can be defined by the unit vectors

$$\begin{aligned} \mathbf{n}_C &= \frac{\mathbf{p}_{1C} - \overline{\mathbf{p}_C}}{\|\mathbf{p}_{1C} - \overline{\mathbf{p}_C}\|}, \\ \mathbf{s}_C &= \frac{\mathbf{p}_{2C} - \mathbf{p}_{3C}}{\|\mathbf{p}_{2C} - \mathbf{p}_{3C}\|}, \\ \mathbf{a}_C &= \mathbf{a}_C \times \mathbf{s}_C. \end{aligned} \qquad (6)$$

Because of the non-homogeneity of the pose vector, one should handle position and orientation errors separately. The position error is evaluated as

$$e_p = \|\mathbf{p} - \overline{\mathbf{p}_C}\|, \qquad (7)$$

where $\mathbf{p}$ is the positional part of the pose used to generate a validation point. The orientation error is evaluated as

$$\mathbf{e}_O = \begin{bmatrix} \arccos(\mathbf{a} \cdot \mathbf{a}_C) \\ \arccos(\mathbf{n} \cdot \mathbf{n}_C) \end{bmatrix} \qquad (5)$$

where $\mathbf{a}$ and $\mathbf{n}$ are the Z and X axes of the top platform frame. We have avoided in this way problems in the error definition due to non-uniqueness of the Euler representation.

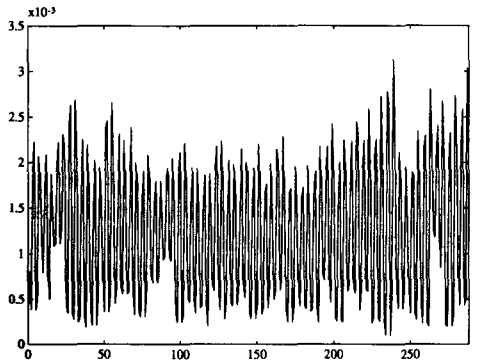

Fig. 4 – Norm of error on elbow joint angles

Fig 5 shows the position errors on the validation set, with a peak value less than 0.2 millimeters. Fig. 6 shows the cartesian orientation error, with a maximum value of $5 \cdot 10^{-3}$ rads.

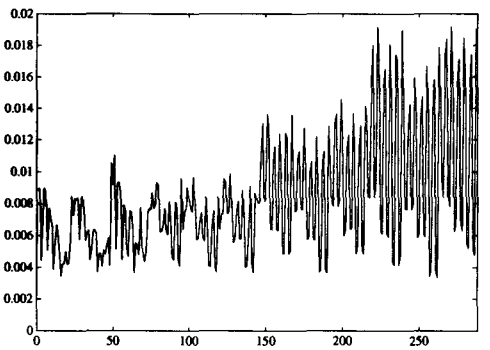

Fig. 5 – Norm of error on cartesian position

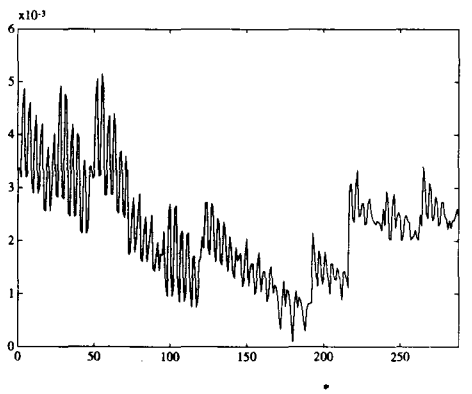

Fig. 6 – Norm of error on cartesian orientation

5 Conclusions

We have proposed a neural-network based method to solve the direct kinematics of a parallel manipulator, a problem which has no closed-form solution. The use of neural networks in this context appears well-founded from a theoretical point of view. We have considered the direct kinematic problem for a parallel manipulator as an approximation scheme of a multivariate function. Our NN solution avoids problems of redundant information by defining the passive joints of the structure as correct network outputs. The unfeasibility of generated elbow angles with respect to the kinematic closure equations has been handled by the proper definition of manipulator pose errors. A novel optimization-based algorithm has been used for training the network, with the aim of achieving a global minimum for the error training criterion. The numerical results are encouraging for the application of this NN solution within a real time motion control law. In order to obtain an approximating scheme highly reliable in a wider workspace, we are currently exploring the use of neural networks derived from regularization theory including Radial Basis Functions [15,16].

References

[1] W.H. Schiffmann and H.W. Geffers, "Adaptive control of dynamic systems by back propagation networks," *Neural Networks*, Vol. 6, pp. 517–524, 1993.

[2] Si-Zhao Quin, Hong-Te Su, and T.J. Mc Avoy, "Comparison of four neural net learning methods for dynamic system identification," *IEEE Trans. on Neural Networks*, Vol. 3, No. 1, pp. 122–130,1992.

[3] T. Yabuta and T. Yamada, "Possibility of neural networks controller for robot manipulator," *IEEE Conf. on Robotics and Automation*, Cincinnati OH, pp. 1686–1693, 1990.

[4] T. Fukuda, T. Shibata, M. Tokita, and T. Mitsuoka, "Neural network applications for robotic motion control; adaption and learning," *Int. Joint Conference of Robotic Manipulator*,San Diego, Ca, pp 447–451, 1990.

[5] M. Kawato Y. Uno, M. Isobe and R. Suzuki, "Hierarchical neural network model for voluntary movement with application to robotics," *IEEE International Conference on Neural Networks*, pp. 573–582, 1987.

[6] A.Guez, Z.Ahmad and J.Selinsky, "The application of neural networks for robotics," in *Neural Networks*, P.G.J. Lisboa (Ed.), 1992.

[7] S. Lee and R.M. Kil, "Inverse mapping of continuous functions using local and global information," *IEEE Trans. on Neural Networks*, Vol. 5, No. 3, pp. 409–422, 1994.

[8] C. Innocenti and V. Parenti-Castelli, "Direct position analysis of the Stewart platform mechanism," *Mech. Mach. Theory*, Vol. 25, No. 6, pp. 611–621, 1990.

[9] A. Kecskeméthy, "On closed form solutions of multiple-loop mechanisms," in *Computational Kinematics*, J Angeles, G. Hommel and P. Kovacs (Eds.), Kluwer Academic Publishers, 1993.

[10] A. Kecskeméthy and M. Hiller, "Automatic closed-form kinematics-solutions for recursive single-loop chains," in *Flexible Mechanisms, Dynamics and Analysis, Proc. of the 22nd Biennal ASME-Mechanisms Conference*, Scottsdale, AZ, pp. 387–393, 1992.

[11] J.P. Merlet "Direct kinematics of parallel manipulators," *IEEE Trans. on Robotics and Automation*, Vol. 9, No. 6, 1993.

[12] K. Hornik, M. Stinchombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, Vol. 2, pp. 359–366, 1989.

[13] K. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, Vol. 2, pp. 183–192, 1989.

[14] V.Ya. Kreinovich, "Arbitrary nonlinearity is sufficient to represent all functions by neural networks: a theorem," *Neural Networks*, Vol. 4, pp. 381–383, 1991.

[15] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, Vol. 78, No. 9, 1990.

[16] T. Poggio and F. Girosi, "Networks and the best approximation property," *Biological Cybernetics*, Vol. 63, pp. 169–176, 1990.

[17] K. Cleary, and T. Brooks, "Kinematic analysis of a novel 6-dof parallel manipulator," *1993 IEEE Int. Conf. on Robotics and Automation*, Atlanta, GA, Vol. 1, pp. 708–713, 1993.

[18] A. Moriconi, "RobLan 2.0 - User manual and implementation notes", Tech. Rep. 02-93, Dip. di Informatica e Sistemistica, Università degli Studi di Roma "La Sapienza", 1993.

[19] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal represention by error backpropagation," in *Parallel Distributed Processing-Explorations in the Microstructure of Cognition*, D.E. Rumelhart, J.L. McClelland and the PDP Research Group (Eds.), MIT press, Cambridge, pp. 318–362, 1986.

[20] L. Grippo, "A class of unconstrained minimization methods for neural network training," *Optimization Methods and Software* Vol. 4, pp. 135–150, 1994.

[21] C.Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks," *IEE Proceedings* Part G, Vol. 139, pp. 301–310, 1992.

[22] L.R. Watrous, "Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimization," *IEEE First Int. Conf. on Neural Networks*, San Diego, CA, Vol. 2, pp. 619–628, 1987.

[23] J. Hertz, A. Krogh and R.G. Palmer, *Introduction to the theory of neural computation*, Addison Wesley, Redwood City, CA, 1991.

[24] P.E. Gill, W. Murray, and M.H. Wright,*Numerical linear algebra and Optimization*, Vol. 1, Addison Wesley, 1991.

Appendix

Let ${}^{\text{leg}_j}\mathbf{p}_j$ be the position vector $\mathbf{p}_{sj}$ of the jth leg tip expressed in the local leg frame. It follows

$$\mathbf{p}_j = {}^{\text{base}}\mathbf{R}_{\text{leg}_j} \cdot {}^{\text{leg}_j}\mathbf{p}_j,$$

where ${}^{\text{base}}\mathbf{R}_{\text{leg}_j}$ is the rotation matrix transforming vectors from the local leg frame to the base frame. Following the notation in Fig. 2, we have

$${}^{\text{base}}\mathbf{R}_{\text{leg}_j} = \text{Rot}\{\bar{\mathbf{z}}, (j-1)\frac{2}{3}\pi\} \cdot \text{Rot}\{\bar{\mathbf{x}}, -\pi\}.$$

Finally, through direct computation

$$\begin{aligned}
{}^{\text{leg}_j}\mathbf{p}_{j_x} &= (L_1 + L_2\text{c}_3)\text{s}_{1j} + L_2\text{c}_{1j}\text{c}_{2j}\text{s}_{3j} \\
{}^{\text{leg}_j}\mathbf{p}_{j_y} &= -(L_1 + L_2\text{c}_{3j})\text{c}_{1j} + L_2\text{s}_{1j}\text{c}_{2j}\text{s}_{3j} \\
{}^{\text{leg}_j}\mathbf{p}_{j_z} &= L_2\text{s}_{2j}\text{s}_{3j},
\end{aligned}$$

where c_{ij} are respectively $\sin(\theta_{ij})$ and $\cos(\theta_{ij})$, while L_i is the length of the th link of each leg and s_{ij}. The actual numerical values are $L_1 = 6.6$ cm and $L_2 = 11.4$ cm.