A Practical Approach for Position Control of a Robotic Manipulator Using a Radial Basis Function Network and a Simple Vision System

Bach H. Dinh, Matthew W. Dunnigan, Donald S. Reay
Electrical, Electronic & Computer Engineering
School of Engineering and Physical Sciences
Heriot-Watt University
EH14 4AS Edinburgh, UK
Email : hbd2@hw.ac.uk

Abstract : This paper proposes a new practical approach using a RBFN (Radial Basis Function Network) to approximate the inverse kinematics function of a robot manipulator. It can be effectively applied for position control of a real robot-vision system in which robot movement in the workspace is observed by a camera. In fact, there are several traditional methods based on the known geometry of the manipulator to determine the relationship between the joint variable space and the world coordinate space. However, these traditional methods are impractical if the manipulator geometry cannot be determined easily, a robot-vision system for example. Therefore, a neural network with its inherent learning ability can be an effective alternative solution for the inverse kinematics problem. In this paper, an approach using a RBFN with predefined centres in the hidden layer (distributed regularly in the workspace) and a combination of the strict interpolation method and the LMS (Least Mean Square) algorithm is presented for effective learning of the inverse kinematic function. By using the strict interpolation method and constrained training data an appropriate approximation of the inverse kinematic function can be produced. However, this solution has the difficulty of how to collect the constrained training patterns whose inputs are selected at pre-defined positions in the workspace. Additionally, the LMS algorithm can incrementally update the linear output-layer weights through an on-line training process. Thus, the proposed idea of combining these techniques can produce the advantages of both methods to deal with the difficulties in practical applications, such as the sensitive structure of a real robot-vision system or a realistic situation where the initial setup and application environments are different. To verify the performance of the proposed approach, practical experiments have been performed using a Mitsubishi PA10-6CE manipulator observed by a webcam. All application programmes, such as robot servo control, neural network, and image processing were written in C/C++ and run in a real-time robotic system. The experimental results prove that the proposed approach is effective.

Keywords: RBFN, robot-vision, strict interpolation, inverse kinematics, PA10-6CE manipulator.

1 Introduction

In robot kinematics there are two important problems, forward and inverse kinematics. Forward kinematics can be regarded as a one-to-one mapping from the joint variable space to the Cartesian coordinate space (world space). From a set of joint angles, forward kinematics determines the corresponding location (position and orientation) of the end-effector. This problem can be easily solved by the 4x4 homogenous transformation matrices using the Denavit & Hartenbergh representation [1][2]. Inverse kinematics is used to compute the corresponding joint angles from location of the end-effector in space. Obviously, inverse kinematics is a more difficult problem than forward kinematics because of its multi-mapping characteristic. There are many solutions to solve the inverse kinematics problem, such as the geometric, algebraic, and numerical iterative methods. In particular, some of the most popular methods are mainly based on inversion of the mapping established between the joint space and the task space by the Jacobian matrix [2]. This solution uses numerical iteration to invert the forward kinematic Jacobian matrix and does not always guarantee to produce all the possible inverse kinematic solutions whilst involving significant computation. In cases where the manipulator geometry cannot be exactly specified, the traditional methods become very difficult, for example a robot-vision system.

The artificial neural network, which has significant flexibility and learning ability, has been used in many robot control problems. In fact, for the inverse kinematics problem several neural network architectures have been used, such as MLPN (Multi-

Layer Perceptron Network), Kohonen self-organizing map and RBFN. In [3][4] Guez et al and Choi described solutions using the MLPN and back propagation training algorithm. Additionally, Watanabe in [5] determined optimal numbers of neurons in a MLPN for approximating the inverse kinematic function. To deal with complex manipulator structures some particular neural network architectures were also presented, for example a combination between the MLPN and the look-up table in [6] or a modular neural network in which the modules were concatenated in a global scheme in order to perform the inverse kinematics in a sequential way in [7]. Similarly, in [8][9][10] the authors proposed using a RBFN to compare with the performance of the MLPN in the inverse kinematic problem. Basically, all of these mentioned approaches used the inverse solution of the forward kinematic transformation to build the mapping from world coordinate space to joint angle space. It means that the manipulator geometry or the forward kinematics must be known to collect the data for training neural networks. Alternatively, when using a simple vision system, the manipulator position in the workspace is represented by an image coordinate in the camera's plane instead of a world coordinate, thus the manipulator geometry is not required. Hence, the RBFN can learn an indirect inverse kinematic function of the manipulator to control its movement in the image plane without any knowledge of the manipulator geometry or forward kinematics. This solution is known as an image-based control scheme. Moreover, an advanced approach using a RBFN with predefined centres in the hidden layer, and a combination of the strict interpolation method and the LMS (Least Mean Square) algorithm is presented to learn the inverse kinematic function. By using the strict interpolation method and constrained training data an appropriate approximation of the inverse kinematic function can be produced. A constrained training set whose inputs are regularly spaced positions represents a different idea from other related papers [8][9][10]. However, this solution has the difficulty of how to collect this constrained data without knowing the inverse kinematic expressions of the real robotic system. Additionally, the RBFN performance can be improved through on-line training using the LMS to incrementally update the linear output-layer weights. Therefore, combining the strict interpolation and the LMS methods produces the advantages of both training methods to deal with the difficulty in collecting training patterns in practical applications. This approach consists of two steps, firstly producing an inaccurate inverse kinematic approximation by the strict interpolation method and then correcting it through on-line training by the LMS. The inaccuracy of the RBFN here reflects the reasonable assumption that the initial network training occurs in an environment that is not exactly the same as the environment where the system is actually deployed. Moreover, it proposes an alternative situation in training neural networks. Instead of learning a function directly from a non-realistic situation, the training process can be divided into two simpler steps. Firstly a less accurate solution is easily obtained producing an approximate function which is then corrected in a convenient way (on-line update in this case) using the adaptive ability of the LMS algorithm. This approach can be used to deal with some practical difficulties in the robot-vision system, such as collecting the constrained training data and the sensitive structure of visual measurement. This is demonstrated by the practical work in Section 3.

The paper is organized as follows. The first section has introduced the basis ideas and background of the inverse kinematics problem using neural networks. In the second section, the approach using the RBFN to approximate the manipulator inverse kinematics is presented. It describes the RBFN architecture and presents the training approach to learn the inverse kinematic function of a robot-vision system. The practical experiment and results are described in the next section that verifies the proposed approach. In the fourth section, aspects of training process are discussed and future research is also identified. Finally, the main conclusions are outlined in the last section.

2 Inverse Kinematics Approximation Using A Radial Basis Function Network

2.1 The structure of Radial Basis Function network

The basic architecture of a RBFN is the three layer network consisting of an input layer, a hidden layer, and a linear output layer [11]. In the inverse kinematics problem of the robot-vision system, the inputs and outputs of the RBFN are position (image coordinates) and joint angles of the manipulator, respectively, shown in Fig. 1.

The unique feature of RBFNs compared to MLPNs and other networks is the process performed at the hidden layer. In this hidden layer, the radial basis function works as a local selector in which the corresponding output depends on the distance between its centre and input. It can be presented as:

$$\Phi_i(x) = \exp\left(-\frac{\|x - c_i\|}{\sigma^2}\right) = \exp\left(-\frac{r_i^2}{\sigma^2}\right) \quad (1)$$

$$r_i = \sqrt{\sum_{k=1}^{K}(x_k - c_{ki})^2} \quad (2)$$

where

Φ

Φ_i – Radial basis function (Gaussian function)

c_i – centre of the i-th hidden unit $(i = 1, 2, \ldots, N)$

σ - width of the Gaussian function

r_i - distance between input and centre of the i-th hidden unit

x – input of the RBFN $(k = 1, 2)$

Thus, the outputs of the RBFN are expressed as :

$$y_j(x) = \sum_{i=1}^{M} W_{ji}\Phi_i(x) \quad (3)$$

where,

W_{ji} – synaptic weight between the i-th hidden unit and the j-th output

y_j – the j-th output $(j = 1, 2)$

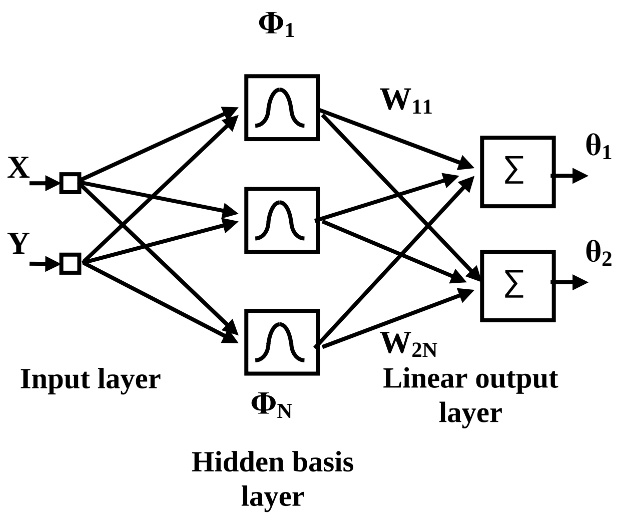


Fig. 1- Structure of RBFN in inverse kinematic approximation of the robot-vision system

Because of the RBFN structure, its training process can be separated into two continuous phases, building a hidden layer structure where centres and widths of the hidden units are determined, and then training the linear weights based on input-output patterns. In order to determine the appropriate centres from the training data set, the first phase could be performed by the self-organizing method, clustering techniques or randomly selected. This is also known as unsupervised learning. The second phase is supervised training to determine the linear weights by either the Least Mean Square algorithm or Gradient Descent method, exactly the same as the back propagation network (MLPN).

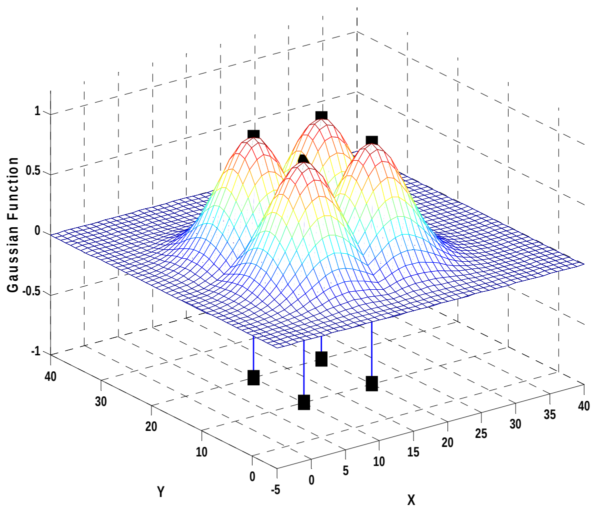


Fig. 2- Centres as regularly spaced positions

In our approach, the structure of the hidden layer is pre-defined by an intuitive method (involving trial and error) where the centres of hidden units are distributed regularly in the workspace and the width of the Gaussian functions are fixed as a proportion of the centre distance. Fig. 2 above shows an example of the hidden layer's centres as regularly spaced positions in the workspace of the manipulator. This is the principal difference from other similar RBFN solutions in [8][9][10] where the centres of hidden units were determined from a particular training set by a clustering technique, and thus that centre set was not fixed, but varied according to the particular training set in the same application. Basically, in our approach the persistent hidden-layer structure is pre-defined and is therefore able to generalize through the whole workspace. These hidden-layer parameters were varied extensively before a satisfactory set was chosen for the specific problem here.

2.2 Supervised learning methods for the linear output-layer weights

As expressed in equations (1) and (3), an output of the RBFN has been performed by a linear combination of the localized bumps formed at the centres of Gaussian hidden units. Hence, this training phase is to determine an appropriate set of the linear weights to be able to approximate the inverse kinematics of a manipulator. It follows and does not depend on the first phase selecting structure of hidden layer above.

In this paper, we propose two different training methods to determine the linear weights for the inverse kinematic approximation. The first one is a simple method where the centres of hidden units coincide with the inputs of training patterns and the linear weights are determined from the target outputs and the interpolation matrix. This is called the strict interpolation method [11] in which the RBFN is an exact mapping of all observations in the training data set.

Given a training data set, consisting of *N* inputs - different positions in the workspace, and *N* targets - corresponding manipulator joint angles: $\{(X_1, X_2, .., X_N); (T_1, T_2, \ldots, T_N)\}$, these known data points X_i are taken to be the centres C_i of the hidden units. Then, after all training patterns have been presented to the RBFN, the *N*x*N* matrix called the interpolation matrix can be obtained as:

$$\Phi = \begin{bmatrix} \Phi_1(1) & \Phi_2(1) & \cdots & \Phi_N(1) \\ \Phi_1(2) & \Phi_2(2) & \cdots & \Phi_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_1(N) & \Phi_2(N) & \cdots & \Phi_N(N) \end{bmatrix} \tag{4}$$

The linear weights are calculated from the targets of training data and the interpolation matrix from the following equation:

$$W = T.\Phi^{-1} \tag{5}$$

This solution is an exact mapping $f : X_i \rightarrow T_i$ $(i = 1,2,\ldots N)$, for all patterns presented to the RBFN and its generalization depends on the appropriateness of the selection of the hidden layer structure (centres and width of Gaussian function). It is an off-line training process and the training patterns are collected before the training phase is performed. In fact, the strict interpolation method is only appropriate when the function to be estimated does not contain high frequency components so that a small number of hidden units for a well-generalized RBFN can be chosen. Fortunately, the inverse kinematic function is a reasonably smooth surface, so the strict interpolation method is a suitable solution if the training patterns are regularly distributed in the workspace. In this paper, the training data used by the strict interpolation method was constrained patterns whose inputs are close to the centres, which were predefined as regularly spaced positions in the workspace. The closer the position of the training data is to the centres, the better the inverse kinematics approximation achieved. This kind of data is called the regularly spaced position training pattern. As a result, this solution can produce an inverse kinematic approximation using a simple structure RBFN with a small number of hidden units [14]. However, this approach has the difficulty of how to collect the constrained patterns without the inverse kinematics expressions. In practice, it can only be estimated by experiment with an undetermined accuracy and also needs many attempts depending on the system's complexity. The quality or accuracy of collected data depends on the user's observation and a poor pattern means that its input deviates from pre-defined position (centres).

The second method called the LMS (Least Mean Square) algorithm is more popular than the former and can be applied for the on-line process. As described in [11], the linear weights can be modified by the simple Delta rule :

$$\begin{aligned} \Delta W_{ji} &= \eta.e_j.\Phi_i \\ e_j &= T_j - y_j \end{aligned} \tag{6}$$

where

e_j - error between target and actual output *j*

Φ_i - output of hidden unit *i*

η – learning rate

This LMS algorithm is quite a simple approach whose performance depends on the adopted learning rate and the size of the training data. Moreover, collecting an arbitrary position pattern is much easier than a constrained pattern as mentioned above. Hence, the related papers [8][9][10] used this arbitrary data (no consideration given to the positions of the patterns) and the LMS algorithm to train the output-layer linear weights. It was the off-line batch training process with data collected based on forward kinematics. However, this approach does guarantee to produce an appropriate approximation function, and also needs many training patterns and training time to cover the entire workspace. In this paper, the LMS algorithm (or Delta rule in incremental case) is used in the on-line process to incrementally update the linear weight by arbitrary patterns. Thus, it should be used as an additional technique to improve the quality of a trained function rather than the main training approach.

Based on the two mentioned learning methods and the predefined structure of hidden layer, a new practical training approach is proposed to cope with the difficulties in approximating the inverse kinematics function in a real robot-vision system, such as collecting the constrained training data and the sensitive visual measurement structure.

2.3 Description of strict interpolation and LMS methods to deal with difficulty in collecting constrained training data

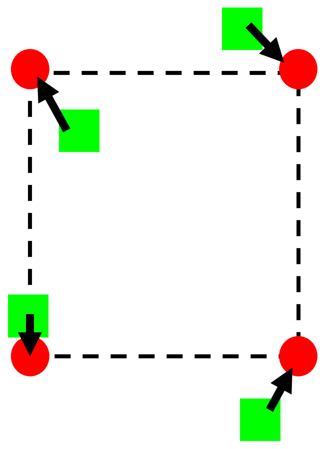
A practical approach for position control of the robot-vision system has been developed to cope with the difficulty in collecting the constrained training data. The notable principle of this approach is that instead of learning a function directly from a complex situation, the training process can be divided into two simpler steps. Firstly using a less accurate but feasible solution to produce an approximate function, and then correcting it through a convenient way (on-line update in this case). The full training procedure can be described as below:

Step 1: Determine hidden layer parameters where
i/ centres of the hidden layer are predefined as regularly spaced positions in the workspace by a specific distance, and
ii/ the width of Gaussian functions is selected as a proportion of the centre distance heuristically.

Step 2: Collect a constrained training set. In the practical work, this can be achieved using the manipulator joint angle controller. This constrained data, consisting of the position and the corresponding joint angles, are picked in the neighbourhood of each centre, but the training data then will be formed as the centres and joint angles of that collected data. Obviously, this is the ***PSEUDO-TARGET*** (inaccurate) training data, whose target outputs do not correspond to the inputs (the centres - regularly-spaced positions). This can be seen in Fig. 3, where the circles and squares are centres and collected points respectively, and thus the inaccurate training set is formed as (*Angle*, *C*).

Step 3 : Use the strict interpolation method and the *pseudo-target* training data to determine the linear weights. Consequently, the currently trained RBFN is an incorrect approximation of the inverse kinematics, but it also produces a boundary function of the accurate approximation trained by accurate constrained training data - (*AngleC, C*) *(if it could be collected)*. That is because both have the same structure and the only deviations are in the corresponding target angles.

Step 4 : Correct the currently trained RBFN by using the LMS (Delta rule) and arbitrary training data. This is on-line training where the linear weights are incrementally updated when a new training pattern is presented.



🟩 ***Actual Position - A***

● ***Centre Position – C***

Actual collected data : (Angle, A)

Accurate constrained data : (AngleC, C)

Pseudo-Target training data : (Angle, C) – Inaccurate (or less accurate)

Fig. 3 – Positions of centres and collected data

Furthermore, ***Step 4*** of the proposed approach can be expanded to deal with the sensitive structure of the robot-vision system, which can occur due to variation in the set-up of the visual measurement system or in case the set-up and actual application environments are different.

3 Practical Work and Results

3.1 Set-up of the practical work

Practical experiments have been developed and performed using the existing facilities in the Intelligent Robotics Laboratory [12]. It is the remotely robotic control system as shown in Fig.4. The following elements of the system can be identified:

- The Mitsubishi PA10-6CE manipulator with servo controller. This is the six link multipurpose arm connected to an industrial PC (IPC) via an ARC-Net interface.
- The IPC running under the QNX Neutrino real-time operating system is used to execute control programmes and communicate with an application PC (APC) via Internet interface.
- A standard webcam mounted on a vertical shaft that permits rotation captures the manipulator images in two dimensional space.

- All of the main application programmes were written in C/C++ and run in the APC.

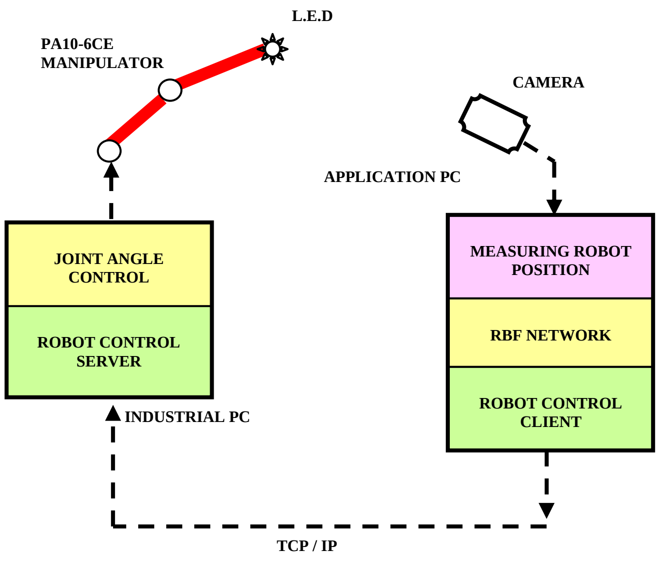


Fig.4 – General structure of the robotic control system

Fig.5 presents a simple robot-vision system, consisting of the PA10-6CE manipulator and the webcam.

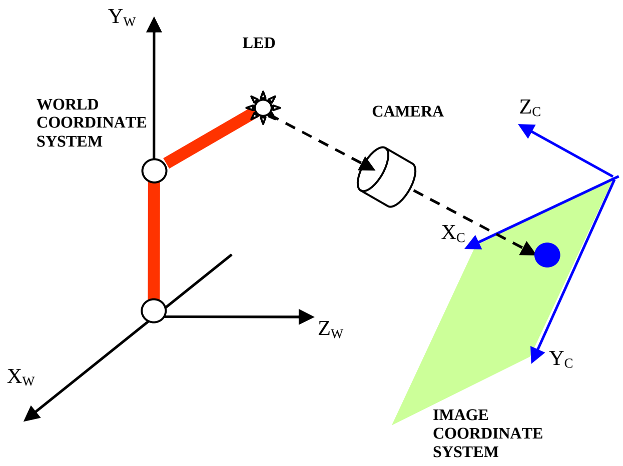


Fig.5 – Simple visual measurement system

The webcam and an image processing programme developed from OpenCV library are used in the simple visual measurement system where the relative position of the manipulator can be directly determined by two dimensional coordinates (*x,y*) in the image plane. Obviously, this can be performed without camera calibration or the manipulator geometry. In other words, the visual measurement system can directly determine the relative positions of any manipulator with unknown geometry in the image plane. Therefore, the inputs of the RBFN are image coordinates (*x,y*) instead of world coordinates (*X,Y,Z*) and the inverse kinematic function then is the mapping from image coordinates (*x,y*) to joint angles. However, this simple visual measurement system is easily altered due to variations in the configured parameters of the robot-vision system. This sensitivity is common in practice.

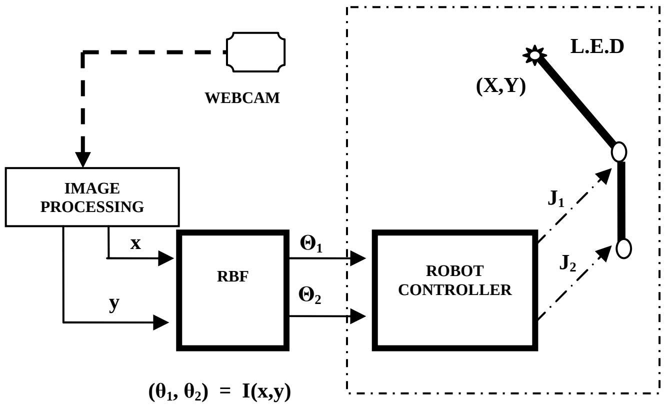


Fig.6 – The inverse kinematics based RBFN for the robot-vision system

Fig.6 presents the general structure of the real robotic system used in this practical work. The PA10-6CE manipulator was controlled to move in two-dimensional space by allowing movement only of the second and third joints (J_2 and J_3). Thus, the problem was to determine the inverse kinematics of a two link manipulator and this RBFN consisted of two inputs (*x,y*) and two outputs (J_2,J_3) correspondingly. In the working area of the manipulator, the centres of the hidden layer were regularly chosen as a grid. The video frame at the resolution of *640x480* (pixels) was created and forwarded to the image processing programme to determine image coordinates (*x,y*) of the manipulator.
This paper presents the practical work through two experiments. The first one is how to train the RBFN to overcome the difficulty of collecting constrained data. The second is the technique to deal with the sensitive structure of the visual measurement system.

3.2 Practical experiment 1

This experiment uses the outlined approach below to overcome the difficulty in collecting constrained training data for the inverse kinematics problem. It can be described in the following steps:

- **<u>Step 1</u> :** A set of regularly spaced position data is manually collected in the workspace of the PA10-6CE manipulator using the joint servo controller and the visual measurement system. The quality or accuracy of collected data depends on the user's observation and a poor pattern means that its input deviates from pre-defined position (centres). However, this step can be simply carried out because it is not necessary to collect the data coincident with predefined positions (centres). To form a set of *pseudo-target* training patterns the inputs are assigned to centres but the target outputs are joint angles corresponding to position of collected points different from the centres.

- **Step 2 :** Using the strict interpolation method and the *pseudo-target* training data to establish an incorrect approximation of the inverse kinematic function. It is the off-line training phase.
- **Step 3 :** this step is used to correct the existing RBFN through an on-line training phase. It can be briefly described as:
    - Moving the PA10-6CE manipulator to an arbitrary position and pick that training pattern.
    - Update the linear weights by the LMS (Delta rule) according to the recent pattern.
    - Repeat until stop command sent.

To verify the performance of the RBFN, a test data set was presented after each training phase. It is a position control task moving the PA10-6CE manipulator along the trajectory as shown in Fig.7, Fig.8, and Fig.9.

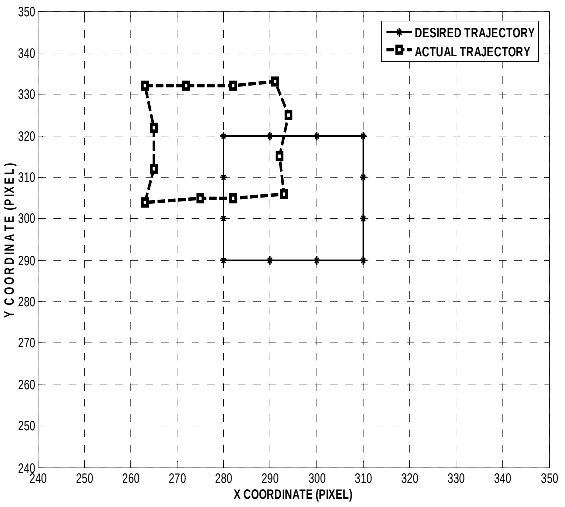


Fig. 7 – Performance of the RBFN trained by pseudo-target (inaccurate) data

An incorrect approximation was built by the strict interpolation method with the *pseudo-target* training patterns as described in S***tep 2***, and the performance of this trained RBFN is presented in Fig.7. The errors introduced are obvious from inspection.

To correct the existing function, an on-line training process was applied using the LMS as described in S***tep 3***. The linear weights are adjusted for each recent training pattern picked from arbitrarily moving the PA10-6CE. After training by a number of training patterns, the RBFN was corrected incrementally and the performance of the robotic system was clearly improved. Fig.8 shows the performance of the RBFN after 10 on-line training patterns.

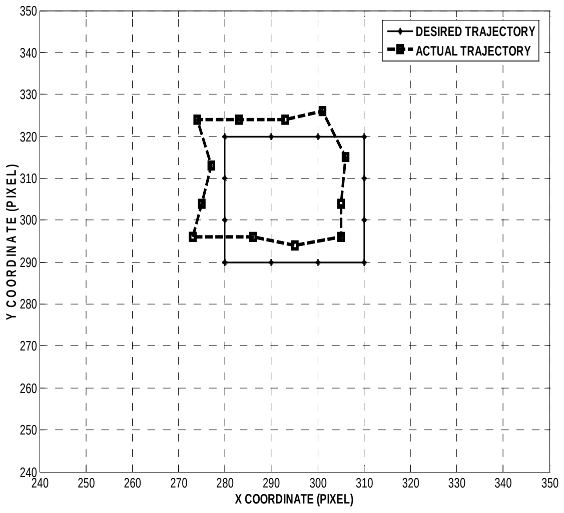


Fig.8 – Performance of the RBFN after on-line training with 10 arbitrary training patterns

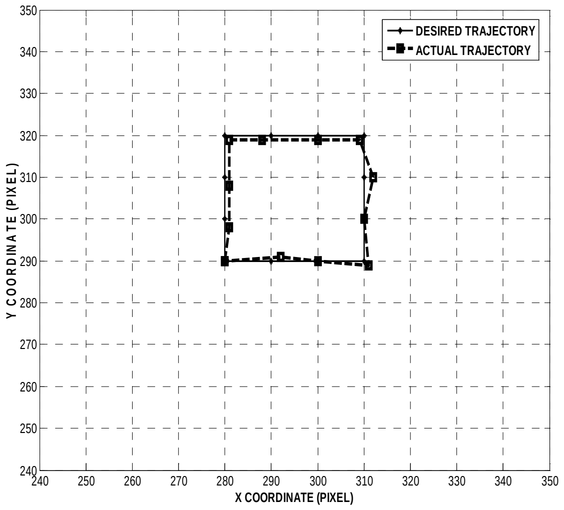


Fig.9 – Performance of the RBFN after on-line retraining with 100 arbitrary training patterns

In Fig.9 the performance of the RBFN after on-line training with 100 patterns is presented. As can be seen, the actual trajectory is close to the desired trajectory with an error of approximately 1 or 2 pixels (one pixel is equivalent to 2 mm). The performance is highly satisfactory because the quality of this robotic system is dependent not only on the RBFN learning ability but also the accuracy of the training data, quality of the visual measurement system, and even the precision of the servo controller. Moreover, the experimental results show that as the distance between the centres becomes smaller the better the generalization. In other words, the RBFN needs more hidden units to improve the accuracy of the inverse kinematic function. Obviously, there is a practical limit in the number of centres of the hidden layer due to lack of memory and complicated architecture of the network.

3.3 Practical experiment 2

This experiment illustrates the effectiveness of the proposed approach in dealing with the sensitive structure of a real robot-vision system. It can be performed similarly to the previous experiments with an additional step as below :

- **Step 1 :** RBFN trained the same as in the previous experiment. This produces an appropriate inverse kinematics approximation of the robot-vision system
- **Step 2 :** If there is any variation in the configuration parameters of the webcam and/or manipulator, the visual structure will be different from the previous case. Consequently, the knowledge of the inverse kinematic function stored in the RBFN no longer matches with the new situation. This reflects the fact that the set-up and actual application environments are sometimes different.
- **Step 3 :** is used to re-correct the existing RBFN by an on-line training phase as was performed in experiment 1.

To alter the set-up of the robotic system, the webcam was rotated through an arbitrary angle so that it changed the image transformation of the visual measurement system correspondingly. Consequently, the inverse kinematic approximation stored in the existing RBFN no longer matches with the new structure of the robot-vision system. Fig.10 presents the performance of the RBFN in the new condition of the visual measurement system and the errors introduced are obvious from inspection.

To obtain a new correct function, an on-line training process was applied using the LMS instead of being retrained by strict interpolation method. In this step, the linear weights are adjusted with each recent training pattern picked from arbitrarily moving the PA10-6CE. After training by a number of training patterns, the RBFN can adapt to the new condition of the visual measurement system and the performance of the robotic system is clearly improved. Fig.11 shows the performance of the RBFN after 60 on-line training patterns.

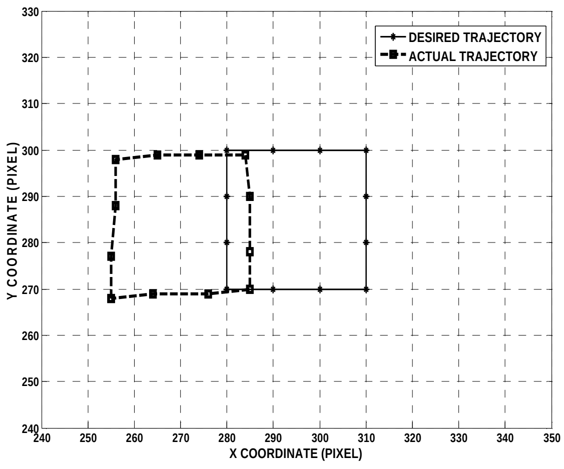


Fig.10 - Performance of the existing RBFN with new condition of the visual measurement system

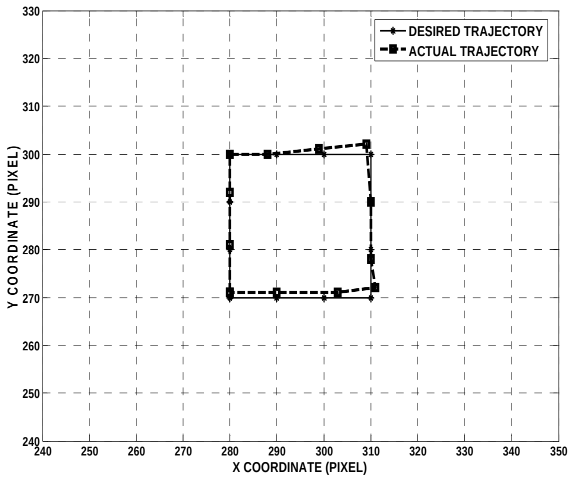


Fig.11 - Performance of the RBFN after on-line retraining with 60 training patterns

4 Discussion

In general, using the pre-defined centres of the hidden layer as regularly-spaced positions means that the hidden layer structure remains the same, and therefore is able to generalize through the whole workspace. As the distance between the centres becomes smaller the better the generalization becomes. Because the inverse kinematic function of the robot-vision system does not contain high frequency components, the strict interpolation method is a suitable solution if the training patterns are collected as the same positions of the centres in the workspace. It can produce a well-generalized RBFN with a small number of hidden units for this inverse kinematics problem. This is a different idea from all other related solutions. However, it also addresses the difficulty of how to collect the constrained data in practice and the proposed training approach is essential to deal with that.

In on-line training, since the RBFN acts as a locally tuned function, only hidden units close enough to the training pattern positions contribute noticeably to the network output. As a result, only the linear weights connected to these hidden units are adjusted via on-line training. It means that the positions of the training data have an important impact in the on-line training process. The closer the training pattern to the test points, the stronger the effect in modifying the linear weights of the RBFN in that area. Different patterns presented to the RBFN can produce different improvement effects in the approximate function. Thus, the distribution of training patterns should cover the entire workspace to modify the whole of the inverse kinematic function.

In addition, the choice of learning rate is an important variable in the LMS algorithm, especially in incremental mode. The variation in learning rate can lead to completely different training results. Therefore, in order to maintain the smoothness of the inverse kinematic approximation over the whole workspace, a small learning rate (η = 0.1) was adopted even though it made the training process slower. This is the reason that the training process required a hundred epochs for the small testing area. In fact, the real number of training patterns for these experiments was collected at only a maximum of 10 points, but they were used repeatedly in the incremental mode with a small learning rate to ensure stability of the learning process. In other experiments, when using the same training patterns with a larger learning rate (0.3 to 0.5 for example), the RBFN could perform well in the testing area after training over a small number of training epochs (from 10 to 20). However, its responses in other neighbouring areas appeared to be worse due to the larger learning rate.

Furthermore, we found that the distribution of training patterns and the learning rate both influence the incremental modification of the linear weights. To keep the training process not only stable but also fast the learning rate should be not a constant but be related to the distribution of training patterns. For example, if a new pattern presented to the network is far away from previous used patterns, the learning rate can be large. In contrast it should be small to keep the training function surface smooth when the current pattern presented is close, or the same, as previous patterns. This aspect requires further investigation.

5 Conclusion

A new approach using a RBFN with predefined centres of the hidden layer, which are distributed regularly in the workspace, and a combination of the strict interpolation method and the LMS (Least Mean Square) algorithm has been proposed to learn the inverse kinematic function. The strict interpolation method with regularly-spaced position training patterns in the workspace can produce an appropriate approximation, but it has the difficulty of how to collect this kind of constrained data. Additionally, the LMS algorithm can incrementally update the linear weights through on-line training process. Therefore, the combination of these techniques can produce the advantages of both training methods to deal with the difficulties in practical applications. Practical work using the PA10-6CE manipulator observed by the webcam verified the proposed approach.

References:

[1] – K.S.Fu, R.C.Gonzalez, C.S.G.Lee. *Robotics : Control, Sensing, Vision, and Intelligence*. McGraw-Hill, 1987.

[2] – W. Khalil and E. Dombre. *Modeling, Identification, and Control of Robots*. Hermes Penton, 2002.

[3] – Benjamin B. Choi and Charles Lawrence. Inverse Kinematics Problem in Robotics Using Neural Networks. *NASA Technical Memorandum, 105869*. Oct.1992

[4] – Allon Guez, Ziauddin Ahmad. Solution to The Inverse Kinematics Problem in Robotics by Neural Networks. *In Proceedings of the IEEE International Conference on Neural Networks*, Vol.1, pp.617-624. July, 24-27 1988. San Diego, California, USA.

[5]- Eiji Watanabe and Hikaru Shimizu. A Study on Generalization Ability of Neural Network for Manipulator Inverse Kinematics. *In Proceedings of the Seventeenth International Conference on Industrial Electronics, Control and Instrumentation*, Vol.2, pp. 957-962. 28 Oct-1 Nov 1991. Kobe, Japan.

[6] – A.S. Morris and A. Mansor. Finding the Inverse Kinematics of Manipulator Arm Using Artificial Neural Network with Lookup Table. *Robotica (Cambridge University Press)*, Vol.15, pp. 617-625. 1997.

[7] – Pablo J. Alsina, Narpat S. Gehlot. Robot Inverse Kinematics: A Modular Neural Network Approach. *In Proceedings of 38th Midwest Symposium on Circuits and Systems*, Vol.2, pp. 631-634. August, 13-16 1995. Rio de Janeiro, Brazil.

[8] – Pei-Yan Zhang, Tian Sheng Lu, Li Bo Song. RBF Networks-Based Inverse Kinematics of 6R Manipulator. *The International Journal of Advanced Manufacturing Technology*, Volume 26, 2004, pp. 144-147. Springer- Verlag London Ltd.

[9] – S.S. Yang, M. Moghavvemi, and John D. Tolman. Modelling of Robot Inverse Kinematics Using Two ANN Paradigms. *In Proceedings of TENCON2000-Intelligent System and Technologies for the New Millennium*, Volume 3, pp. 173-177. September, 24-27 2000. Kuala Lumpur, Malaysia.

[10] – Joseph A. Driscoll. Comparison of Neural Network Architectures for the Modelling of Robot Inverse Kinematics. *In Proceedings of The IEEE SOUTHEASTCON 2000*, Vol.3, pp. 44-51. April, 7-9 2000. Tennessee USA.

[11] – Simon Haykin. *Neural networks - A Comprehensive Foundation*. Prentice Hall, Inc. 1999.

[12]- Cyprian M. Wronka, Matthew W. Dunnigan. Internet Remote Control Interface for A Multipurpose Robotic Arm. *The International Journal Of Advanced Robotic Systems.* Volume 3, pp. 179-182. June 2006.

[13] – OpenCV: Image processing and computer vision reference Manual.

[14] - Bach H. Dinh, Matthew W. Dunnigan, Donald S. Reay. Position control of a Mitsubishi PA10-6CE manipulator using the RBFN for inverse kinematics approximation. *In Proceedings of the International Conference on Robotics, Vision, Information, and Signal Processing ROVISP 2007*, pp.170-174. Penang, Malaysia, November 28-30, 2007.