Neural Comput & Applic (1994)2:2–12
© 1994 Springer-Verlag London Limited

Neural Computing & Applications

Using Neural Networks to Learn Hand–Eye Co-ordination

Marggie Jones and David Vernon

Department of Computer Science, Trinity College, Dublin, Ireland

The work presented in this paper shows how the association of proprioceptive and exteroceptive stimuli can enable a Kohonen neural network, controlling a robot arm, to learn hand–eye co-ordination so that the arm can reach for and track a visually presented target. The approach presented in this work assumes no a priori *model of arm kinematics or of the imaging characteristics of the cameras. No explicit representation, such as homogeneous transformations, is used for the specification of robot pose, and camera calibration and triangulation are done implicitly as the system adapts and learns its hand–eye co-ordination by experience. This research is validated on physical devices and not by simulation.*

Keywords: Hand–eye co-ordination; Kohones network; Neural networks

1. Introduction

One of the first skills which humans learn is how to control their limbs. The aim of the work presented in this paper is to show how the association of proprioceptive and exteroceptive stimuli can enable a neural network, controlling a robot arm, to learn hand–eye coordination so that the arm can reach for and track a visually-presented target.

The inverse kinematic problem is that of computing robot arm joint positions for a given position and orientation of the end-effector [1], i.e. it involves a mapping from x,y,z Cartesian space to $\theta_1,\theta_2, \ldots \theta_n$ joint space. The hand–eye problem requires another stage: the mapping from camera (eye) image plane coordinates to spatial coordinates. Taking the stages together then, the hand–eye problem involves a mapping from i,j image space to x,y,z Cartesian space followed by a mapping from Cartesian space to $\theta_1,\theta_2, \ldots \theta_n$ joint space. The first mapping stage is traditionally done using triangulation methods while, for the second stage homogeneous transformation matrices are often used to represent the x,y,z roll, pitch and yaw configuration of the manipulator, i.e. its pose in Cartesian space [2, 3]. The mapping itself from Cartesian space to joint space is normally achieved by analytic mathematics. As well as using such homogeneous transformations and triangulation methods, details about the cameras, in the form of camera calibration models, must be included to fully specify the system. If a camera position or the length of one of the robot arm segments are changed, the calculations just detailed must be redone to find the inverse kinematic solution for this new configuration. Such traditional methods can be complex and cumbersome, and are further complicated by the fact that a given spatial position can be achieved by the robot arm's end-effector using several different motor position configurations.

In the work presented in this paper, in which a neural network controlling a robot arm learns hand–eye coordination, the inverse kinematic problem *per se* is not solved. This is because in this work, camera plane images (i.e. extra-body or exteroceptive input stimuli) and motor position information (i.e. body or proprioceptive input stimuli) are used, but real-world x,y,z spatial coordinates are never explicitly referred to or calculated, i.e. the mapping developed in this system (learned by the neural network by association) is

Original manuscript received 28 October 1993

Correspondence and offprint requests to: M. Jones, Management Information Systems, Department of Accountancy and Finance, University College, Galway, Ireland.

Neural Comput & Applic (1994)2:1
© 1994 Springer-Verlag London Limited

Neural Computing & Applications

Editorial

An applications journal has a responsibility to present papers which are of current interest both in academic and direct application environments. I believe that the four papers presented in this issue cover areas of acute current interest: robotics, finance, rule extraction, and data analysis.

In the field of robotics, neural networks are rapidly assuming importance due to their adaptive abilities, with supervised and unsupervised learning algorithms. Research interest is focusing very strongly on the self organising algorithms, and the first paper presents an interesting practical investigation into the subject, with many ideas which are widely applicable.

The second paper expands on the work presented by the same authors in the very first issue of the journal. The principles of asset location, foreign exchange, and bond trading are considered, all of which being matters of concern and interest at the present time. Certainly in the UK, there has been an enthusiastic take-up of neural computing in companies in the financial sector. Understandably, they are not always ready to broadcast their expertise, so we are pleased to be able to review some of the techniques used in this rapidly developing and lucrative field.

Newcomers to neural network practice are frequently concerned about the relationship between rules and the heuristic nature of neural network learning. The third paper looks at the rule problem and, using diagnosis as a testbed for the experiments, addresses the question of the interface between numerical and symbolic knowledge representations and their integration.

Market research is a fertile area for the application of neural computing. The question is how to overcome the practical difficulties, and the final paper looks at data derived from interview forms. Various problems of identifying trends and interpreting unsupervised learning maps were studied by the authors, who present a vector quantisation technique to identify trends and clusters. The paper brings into relief the problems of using neural networks to analyse questionnaires, and contributes some interesting methods to this area of market research practice.

Howard James
Editor

from i,j image space to $\theta_1,\theta_2, \ldots \theta_n$ joint space. The final result, however, is the same in both the traditional method and in this work, i.e. a set of motor positions, which will result in the robot arm's end-effector being at a target position, is computed. The approach presented in this work assumes no *a priori* model of arm kinematics or of the imaging characteristics of the cameras. Homogeneous transformations are not used. There is no need to produce a camera calibration model, nor to do explicit triangulation, because these are effected automatically as the system adapts and learns to associate the position of a target, as presented visually to two spatially-separated cameras, and thus characterised by two pairs of image co-ordinates (i_1,j_1) and (i_2,j_2), with the desired joint coordinates $(\theta_1,\theta_2,\theta_3)$. The image coordinates represent the exteroceptive information, while the joint coordinates represent the proprioceptive information.

The inspiration for the approach taken in this work was the fact that epigenetic structures are known to exist in the human brain, i.e. structures that develop after birth, so that, for example, when a baby waves its arm about in a seemingly random fashion it is, quite literally, developing hand–eye co-ordination and learning by experience as it superimposes epigenetic structures onto its genetic brain structure. In doing this, a baby makes use of sensory feedback from its eyes (exteroceptive information) and its limbs (proprioceptive information). Movements directed towards external objects are generally organised on the basis of several types of sensory signals [4]. But the key to producing hand–eye co-ordination is the fact that these signals must be integrated in a meaningful way before the required hand–eye co-ordination can be achieved [4, 5].

A further statement of the necessity for the correct integration of the different types of sensory modalities is given by Held and Hein [6], who have shown that in kittens, if vision and proprioception are experimentally dissociated, sensorimotor co-ordination cannot be learned.

In the work presented in this paper, proprioceptive and exteroceptive inputs are integrated in a meaningful way to produce the required hand–eye co-ordination, where an adaptive and to some extent neuromimetic neural network learning approach was adopted. A robot arm's end-effector was moved many times, in a random manner, within a cubic space. For each step of a learning or training phase, two cameras focused on an infra-red LED (which will later act as the target to be reached for) placed in the robot arm's end-effector, and they produced co-ordinates of its whereabouts in their image planes. These co-ordinates served as exteroceptive inputs to the neural network, while the robot's motor positions served as proprioceptive inputs. These two types of sensory modalities were associated in a mapping which evolved in the neural network's weights. The random movements of the arm are important in the development of sensory-motor coordination because it should lead to uniform sampling of the input space, and hence to an unbiased ability to reach for the target [6]. After training, the robot arm was able to reach for a visually presented target which was in the field of view of the two cameras and the spatial envelope within which the robot arm was randomly moved during training.

Other neural network approaches to this problem have been taken. They include work by Jordan [7], Massone and Bizzi [8], Dean [9] and Mel [10, 11]. The work of the last author also caters for obstacle avoidance and path planning. All of these approaches use a supervised learning method during the training phase, where for each input the network was told what its output should be. Unsupervised learning neural network approaches (where desired – or target – outputs which are specified *a priori* are not used in the training phase) include those by Kuperstein [12], Coiton [4], Ritter *et al.* [13–15] and Walter and Schulten [16]. Of these, only Kuperstein's did not use the Kohonen network paradigm. Using Kohonen's paradigm, one can produce self-organized topological mappings of the input, i.e. mappings such that input points which are close together are represented in the mapping by points which are close together.

It is these topological mappings which are crucial to the task in hand, because they associate the two types of sensory input. There is an implicit assumption in this work that such a mapping exists, an assumption also made by others who use neural networks for the task of hand–eye coordination [4, 13–15]. It is not, however, assumed that such a mapping is a bijection; indeed, a given spatial position can be achieved by the robot arm's end-effector using several different motor configurations. It is important to note also that these mappings do not provide us merely with a massive look-up table that links motor and spatial positions. This is because neural networks can generalise to deal with inputs which they have not encountered before (but which are similar to ones with which they have been presented). Kohonen networks are ideally suited to producing such topological mappings with this ability to generalise. If, however, another method of producing such mappings could be found it would be just as useful for this hand–eye coordination task.

The work of Ritter *et al.* involves a simulation which uses an extension of the Kohonen paradigm along with an error correction scheme based on the Widrow–Hoff learning rule to learn hand–eye co-ordination of a simulated 3-link robot arm. During training of his neural network, the target to be reached for is not placed in the robot arm's gripper. Instead it is randomly positioned in space in front of the robot arm. Two cameras are used to locate the target in their image planes and this alone serves as input to the neural network, i.e. correlated proprioceptive and exteroceptive inputs are not used. The arm, governed by the network, makes an attempt to reach for the target and the error in resulting arm position relative to the target is corrected. Ritter's work used a 3D neural network to match the 3D work space. Three degrees of freedom of the simulated arm were also employed. On the other hand, the research presented in this paper involves validation on physical machinery rather than in a simulated environment.

As in the work presented in this paper, Coiton built up a correlation of two types of sensory input on a Kohonen network layer by placing an object between a robot arm's grippers and moving it randomly in space. This object later became the target to be reached for during testing. Both the work presented in this paper and Coiton's work involve a mapping from one space to another, the former work involving a mapping from image space to joint space, with the latter work involving a mapping from Cartesian space to joint space. Joint space then is involved in the mapping for both types of work. The significant differences between Coiton's work and the work presented here are:

- Instead of using two cameras to collect exteroceptive inputs and associating stereoscopic image plane data with robot arm joint values, i.e. true hand–eye co-ordination which inherently comprises both triangulation and the inverse kinematic solution, Coiton *et al.* used three orthogonal sensors to produce x,y,z Cartesian space co-ordinates of the object/target position. Coiton's network thus associates Cartesian space inputs with robot arm motor angles; i.e. his Kohonen network solves the inverse kinematic problem, it does not produce hand–eye co-ordination.
- A consequence of this is that Coiton *et al.* had six inputs into their Kohonen network, while seven were used in the system described in this paper, i.e. Coiton used the same numbers of proprioceptive and exteroceptive inputs while this work used a larger number of exteroceptive inputs.
- Coiton *et al.* updated their network weights in a different way to that used in this paper. They used a Gaussian neighbourhood function and updated close-by neighbouring cells to the winning cell in an excitatory fashion, while more distant cells were updated in an inhibatory manner. This gain and neighbourhood values were decreased over time but not linearly. In contrast, a simple square neighbourhood was used in the work presented in this paper. No inhibition was used in updating more distant cells, and both gain and neighbourhood values were decreased linearly.

More recently, Walter and Schulten [16] have investigated the use of a modified version of the Kohonen paradigm to effect servo-control of a PUMA 562 robot. As in the system described in this paper, Walter's and Schulten's system utilises two cameras. In this instance, however, the proprioceptive information about the joint positions is not input to the network. Instead, the system depends solely on the exteroceptive data constituted by the image co-ordinates of the target position and the current position of the end-effector. Furthermore, the system operates by implementing a closed-loop control architecture whereby the image co-ordinates of both target and end-effector are computed for each individual movement in the total servo-motion in an effort to eliminate the difference in their positions. This contrasts with the approach in this paper, which is more concerned with the association of proprioceptive and exteroceptive data; proprioceptive data identifying the current position of the end-effector and exteroceptive image co-ordinates identifying only the target position.

In the following, we will first describe the experimental set-up, i.e. the equipment which was used to validate the work and the essential organisation of inputs and outputs, and then we will detail the approach which was used to produce hand–eye co-ordination from this system. This is followed by an analysis of the system's performance and we then proceed to draw some general conclusions about the work.

2. Experimental Set-up

2.1. The Hardware

Three essential components are required to effect hand–eye coordination: a robot arm (the 'hand'), a camera and image acquisition system (the 'eye'), and a computer system which implements the Kohonen network. In the system described in this paper, a Universal Machine Intelligence (UMI) RTX SCARA-configuration robot arm is used. This arm has six degrees of freedom: one translational (prismatic) motion of the arm in the vertical z direction, a rotational (revolute) motion of the shoulder, θ_s, a rotational (revolute) motion of the elbow, θ_e, and roll, pitch and yaw wrist movements (see Fig. 1).

In the work described in this paper, only the first three degrees of freedom are used. The RTX is a low-cost robot rather than an industrial robot and has a nominal positional repeatability of 0.5 mm at the wrist with the arm at its full extension of 507 mm in the 'straight out' position. For the training phase, an infra-red light emitting diode (LED) is placed between the robot arm's grippers. This LED is viewed alternately by two stationary filtered cameras which can detect only infra-red light.[1] Both images are transferred via an Imaging Technology framestore board to the controlling computer's memory where, using a simple computer vision thresholding algorithm, the i,j positions of the LED's centre in the cameras' image planes are computed. These positions serve as exteroceptive inputs to the neural network. The computer also controls the robot arm movements, and can communicate with it to determine its current motor

[1] The LED is used simply to ease the task of visual identification of the position of the arm; it renders trivial the normally difficult process of segmentation (differentiating between object of interest – the arm – and the background) since only the infra-red light emitted by the LED is imaged by the cameras.

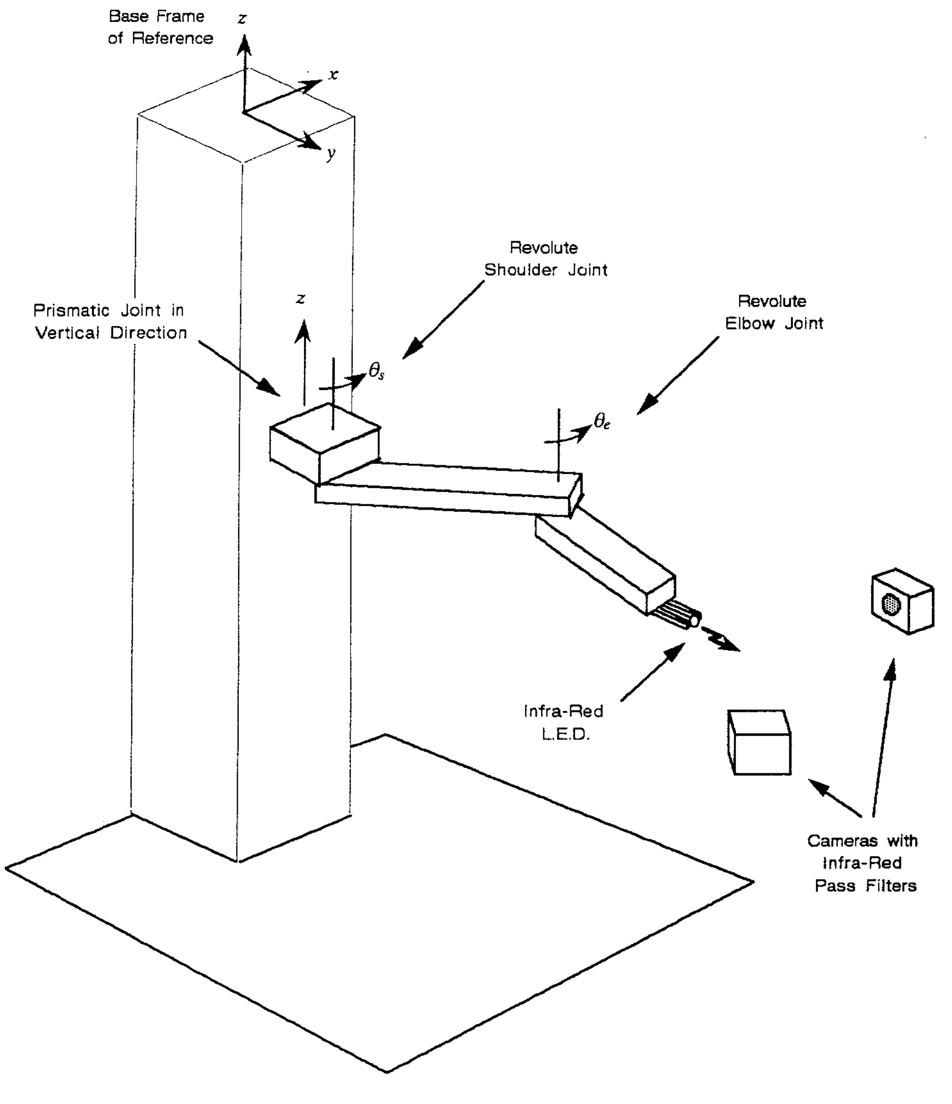

Fig. 1. Schematic diagram for the experimental layout.

positions. These positions serve as proprioceptive inputs to the neural network.

2.2. The Software

Although a real robot arm is used in this work so that robot arm movements and positions are not simulated, the architecture and mechanism for the Kohonen artificial neural network paradigm are implemented in software. The network is configured as follows (refer to Fig. 2).

Exteroceptive inputs are given by the two cameras' i,j image plane co-ordinates of the position of the LED (i_1,j_1,i_2,j_2), while proprioceptive inputs are given by the corresponding vertical translation, shoulder and elbow robot arm motor positions (z,θ_s,θ_e) which cause the LED to be in that particular position. The input layer to this Kohonen network then is comprised of seven sensory inputs $(i_1,j_1,i_2,j_2,z,\theta_s,\theta_e)$. Each of these inputs has a weighted connection to every node, or unit, in the second layer. These weighted connections are called the *sensory weights*. The weights associated with the proprioceptive inputs will later also be used to command the robot arm to move to different configurations when the neural network, controlling the robot arm, has been trained. Details of how these inputs are generated will be given in the next section.

3. Network Training and Verification

3.1. Training

The steps involved in training a Kohonen network are as follows:

- Randomly initialize all weights to be in the range zero to one. Then repeat the following steps for T iterations, where T is the number of training steps.
- Randomly choose inputs (i.e. a robot arm configuration) in the range zero to one. (This step is described more fully later in this section.)
- On the basis of a Euclidean distance metric, find the node whose weights are most similar to the input.
- Update that node's weights and those of its neighbours according to the following equations:

$$[wt_{ji} = \alpha(ip_i - wt_{ji})] \tag{1}$$

$$[wt_{jinew} = wt_{jiold} + wt_{ji}] \tag{2}$$

  where wt_{ji} is the weighting between node j and input i, ip_i is the i-th input, and α is the gain or learning rate (an adjustible parameter with value less than one).
- Reduce neighbourhood size and learning rate as per the following two equations:

$$\left[d = d_0\left(1 - \frac{t}{T}\right)\right] \tag{3}$$

  where d_0 is the initial neighbourhood size, t is

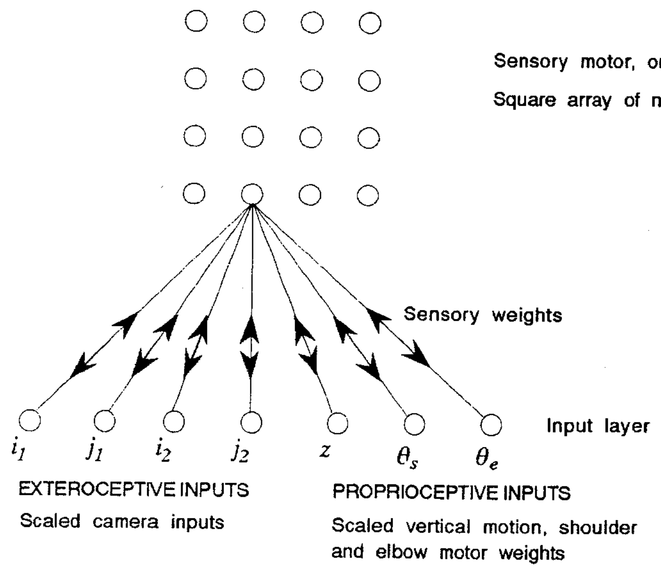

Fig. 2. Architecture of the neural network which learns the hand–eye co-ordination for the given robot arm and camera configuration.

the current updating/training iteration, and T is the total number of iterations:

$$\left[\alpha = \alpha_0\left(1 - \frac{t}{T}\right)\right] \tag{4}$$

Typically, the neighbourhood size begins large, e.g. one-half to one-third of the grid size. In this instance, a simple square neighbourhood is used and neighbourhood size can only have integer values.

Altering α_0 can speed up or slow down how quickly the network organises. For this experiment it was empirically chosen to be 0.5.

From equation (1) it can be seen that the updating process stops automatically when α becomes zero. It should also be stopped when the neighbourhood size reaches zero.

For the task of hand–eye co-ordination, the random inputs in the range zero to one are found in the following way. Random values for the vertical translation, shoulder and elbow motor positions, z,θ_s,θ_e, are chosen. If these positions would result in the robot arm's end-effector being within a pre-defined rectanguloid space, centred in front of the arm's torso, then the arm is commanded to move to this motor configuration. If the random values would not result in such a configuration, then another set of random values are chosen, and this is repeated until a set of motor positions satisfying the (unchanged) rectanguloid space condition is satisfied. The two cameras then view the infra-red emitting LED, which is between the arm's grippers, and the i,j position of its centre in their image planes is found. If either of the cameras is unable to view the LED, then another random position, satisfying the rectanguloid space condition must be found. The seven inputs, i.e. three motor positions (z,θ_s,θ_e) and four camera image plane positions (i_1,j_1,i_2,j_2), are scaled to be in the range zero to one, and are then fed as inputs to the input layer of the network. All seven inputs are scaled so that each will contribute evenly to the evolving network's distribution of weights.

3.2. Testing

When training is completed the system is ready to be tested. The LED is removed from the robot arm's grippers and is placed somewhere within the predefined space, in front of the robot arm, within which the neural network was trained. It must be placed in such a manner that it is visible to both cameras. The LED, thus positioned, has become the target for which the robot will reach. The i,j co-ordinates of the LED's centre in the camera image planes are found once again. The controlling computer interrogates the robot arm and finds its current vertical translation, shoulder and elbow positions (z,θ_s,θ_e). Scaled versions of all seven inputs are fed into the trained neural network. A Euclidean distance competition is held for the seven inputs and a winning node is found. The weights corresponding to the proprioceptive inputs attached to this winning node, when scaled up to their full range values, should produce a set of motor positions, which should result in the robot arm's end-effector being closer to the target. This process is then repeated, i.e. the scaled-down camera inputs (which should be the same as before if the target is stationary) and the scaled-down versions of the robot arm's new motor positions are fed into the trained net, a winning node is found and scaled-up versions of its weights corresponding to the proprioceptive inputs are used to command the robot arm to move to a new position which should be even closer to the target. This process is continued until the latest position to which the robot arm is moved is the same as the previous position, i.e. no more movements are required. Thus for a stationary target the robot arm should home in on it, initially making large movements and then smaller and smaller ones, i.e. the system should act as a servomechanism. For a moving target the robot arm, controlled by the neural network, should be able to follow it and track its movements.

3.3. Underlying Assumption

Before proceeding further, it is worth considering why the Kohonen network is a suitable mechanism for servo-control, and how servo-control can be effected by such a network. We first note that proprioceptive and exteroceptive inputs are correlated during training, in the sense that given motor inputs cause the LED which is grasped by the robot gripper to be in a certain position which is detected by the cameras and fed to the network as exteroceptive inputs. Thus, during training, the inputs are co-ordinated hand–eye inputs.

In the testing phase, the inputs are no longer correlated. Proprioceptive and exteroceptive inputs which belong to different correlation groups are used as inputs to the network. Were the proprioceptive inputs, found during the testing phase, matched with their corresponding correlated exteroceptive inputs (from the training phase) and used as inputs to the network, they may have caused one node in

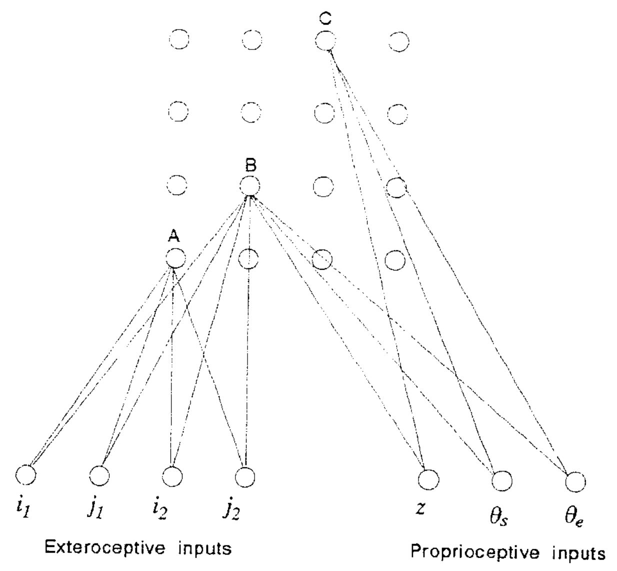

Fig. 3. Node A corresponds to a possible winning node associated with inputs that are correlated with training-phase exteroceptive inputs; node B corresponds to a possible winning node associated with uncorrelated inputs; node C corresponds to a possible winning node associated with inputs that are correlated with training-phase proprioceptive inputs; the actual winning node corresponds to a motor arm position part-ways between the current robot arm position and the target position.

the sensori-motor layer to have won the Euclidean distance competition. If the exteroceptive inputs, found during the testing phase, were matched with their corresponding correlated proprioceptive inputs (from the training phase) and used as inputs to the network they may have caused a different node to have won the Euclidean distance competition. Because Kohonen networks form topological mappings of the inputs space, so that two similar input points cause nodes which are close to each other in the output layer to win the Euclidean distance competition, it is anticipated that using uncorrelated inputs during the testing phase, as just described, would result in a winning node that would correspond to a motor arm position part-ways between the two positions to which the two correlated sets of inputs correspond, i.e. part-ways between the current robot arm position and the target position (see Fig. 3). This assumption underlies the work described in this paper, and it is one of the primary goals of the paper to validate this assumption.

4. Validation of the System

The important parameters associated with the training of this system are:

- the number of training iterations
- the array size
- the initial neighbourhood size
- the initial learning rate/gain
- the time to complete training.

The dimensions of the working space over which the system was trained are 35 cm in the z (height) dimension, ranging (in the robot's co-ordinate frame of reference) from -55 cm to -20 cm; 34 cm in the x (width) dimension, ranging from -17 cm to 17 cm; and 9 cm in the y (depth) dimension, ranging from 39.5 cm to 48.5 cm. These x and y values were chosen because they provided an area over which the end-effector's movements could be evenly spread, thus producing an even distribution in network weights.

Three distinct scenarios were investigated: a 4×4 array size, with an initial neighbourhood size of 2; a 10×10 array size, with an initial neighbourhood size of 4; and a 20×20 array size, with an initial neighbourhood size of 8. In each case, the initial learning rate was set at 0.5. The number of training iterations for the 4×4, 10×10, and 20×20 array size was 1000, 1500, and 6000, respectively. It should be noted that the 4×4 array size was used primarily to verify that the system was functioning correctly; it does not represent a plausible network for robot control.

In all three scenarios, the network enabled the end-effector to reach for the visually presented target, iteratively reducing the distance between the end-effector and the target, initially by making gross movements and then by making finer ones. Figure 4 depicts a graph of the average length of a movement (in a sequence of movements which are made in attempting to reach for a given target) as a function of the number of the movement in the sequence.

Some points are worth noting about the behaviour of the different systems. First, larger distances between the initial position of the end-effector and the target required more individual movements in order to achieve the target position. Second, networks comprising a larger number of nodes (e.g. 20×20 array vs. 10×10 array) also effect the total movement using a larger number of individual movements. This is so because a node in Kohonen networks which use larger arrays corresponds to a smaller volume, or receptive field, in the working envelope of the robot than a node in a smaller array. Thus, nodes in larger arrays are capable of discriminating between two similar inputs which would have caused the same node to win in smaller arrays. Because of this, while the network 'decides' that it has reached the target in the case of a

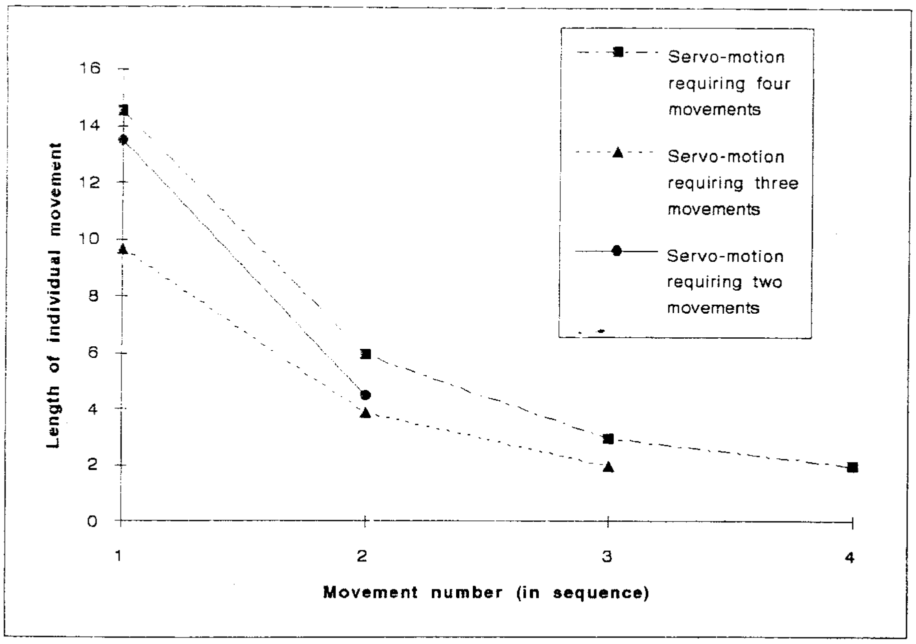

Fig. 4. Average length of movement (in a sequence of movements which are made in attempting to reach for a given target) as a function of the number of the movement in the sequence.

smaller network, i.e. the current winning node is the same as the last winning node, based on similar inputs, so that no more movements are required for it to reach closer to the target, a different node will win in the larger network, based on the same similar inputs, indicating that another movement is necessary.

This also explains why using a larger network results in the end-effector being closer to the target after the sequence of servo-movements than is the case for a smaller network. For the larger network, the average distance between target and robot arm end-effector (once the arm had reached for the target) is 4.51 cm for targets at a height of −55 cm, 3.62 cm for target heights of −40 cm, while for targets at −25 cm it is 3.65 cm. These values are smaller than for corresponding heights using the smaller array/fewer training steps combination. The closest distance which the end-effector achieves in reaching for the target during testing was 1.28 cm. Occasionally, the distances between the target and robot arm which result after reaching movements have been made are quite large, i.e. approximately 6 cm. It is unclear why such outliers occur.

A boundary effect can be seen in this (larger) network, i.e. a diminished ability to reach for the target at the edge of the working space over which the network was trained. This can be seen from the fact that the resulting Euclidean distance between target and end-effector was larger for targets placed at the lower extreme of the working space than for those in the middle of it. That a boundary effect exists is also borne out on examination of z value ranges reached by the end-effector at different target heights. For target heights at −25 cm and −40 cm, the arm height ranges straddle the target heights, while for target heights at −55 cm they do not. This boundary effect was larger for the system resulting from the smaller array/fewer training steps combination. Such boundary effects have been discussed extensively by Kohonen [17].

Tests were also conducted on the 10 × 10 system with targets placed at random x and y positions, with the value of z fixed at −25 cm. In this configuration, the system was unable to reach for the desired targets. This occurs because the selection of random training positions is biased towards the lower end of the z range, i.e. −55 cm, and against the upper end, i.e. −20 cm. Hence the system is better able to learn, and subsequently perform, at this height. This can be seen from Fig. 5, which depicts the spatial points used during training. The question then arises, of course, as to why there is a bias in these randomly selected training points. The reason is that, although the randomly chosen spatial positions were deemed to be acceptable training positions if they would result in the end-effector being within the pre-defined training envelope, there is an additional constraint that both of the cameras must be able to locate the LED in their field of view at that position. If they cannot, then that position is rejected, i.e. it is deemed not

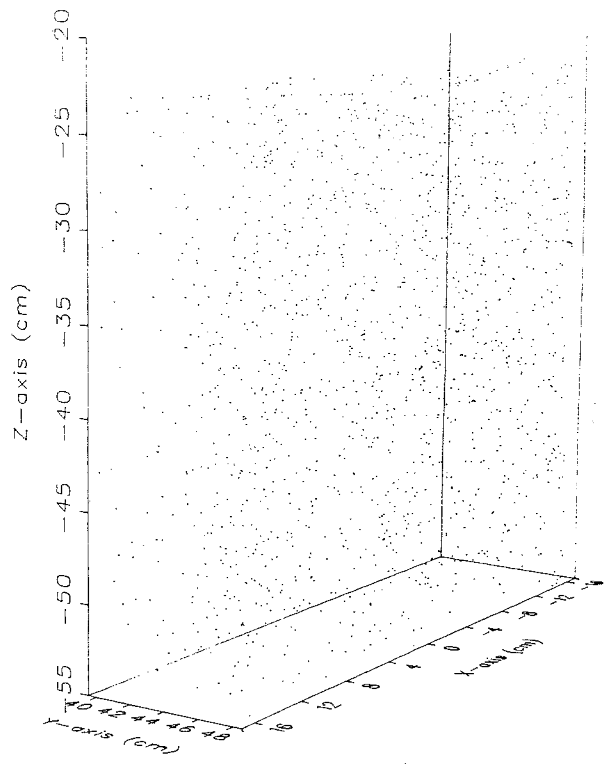

Fig. 5. Distribution of points used when training the 10 × 10 network.

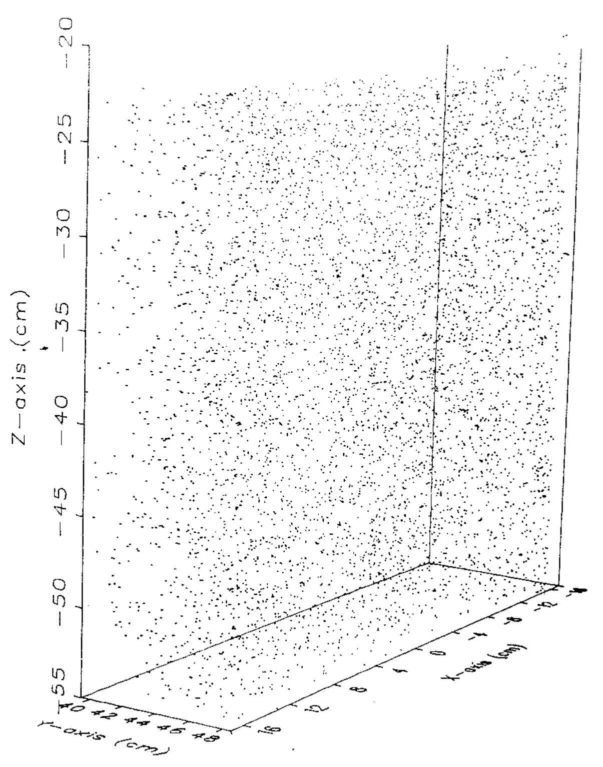

Fig. 6. Distribution of points used when training the 20 × 20 network.

to be a suitable training point. This happens when selecting points in the upper part of the working envelope (i.e. towards the end with values of −25 cm in the *z* dimension). This results in a bias in the training positions, and the inability of the system to reach for points in the upper part of the training envelope.

Figure 6 depicts the distribution of training positions which were used for the larger test array (20 × 20 nodes). A bias once again clearly exists in this set. It can also be seen from this figure that the training area was both densely and uniformly sampled below the −25 cm level.

Apart from the boundary effect and this camera-induced bias, neither the 10 × 10 system nor the 20 × 20 system displays any significant non-uniformity in accuracy as a function of position in the work-space.

In principle, the system is capable of tracking a moving target, i.e. if the target was moved from one position to another after the robot arm had made some movements it could then move to the new target position, homing in on it as before. However, because the robot arm moves quite slowly it could not track a target in real-time. It is the arm movements themselves, and not the software governing the system, that is the limiting factor in this case.

Figure 7 depicts a three-dimensional plot of the *x*,*y*,*z* values corresponding to the network nodes' weight values where neighbouring array nodes have been linked. It resembles a corrugated or ruffled plane which does not completely cover the training space and represents the system's attempt to map a higher dimensional problem space onto a two dimensional plane.

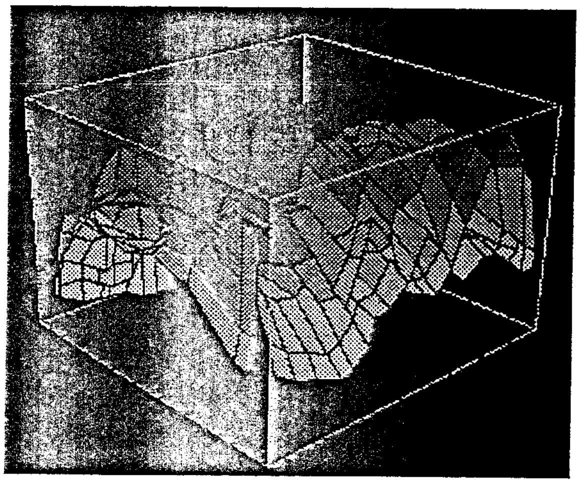

Fig. 7. Three-dimensional plot of the *x*, *y*, *z* points corresponding to the network nodes' weight values where neighbouring array nodes have been linked.

Another point of interest about the result from this training run is the variation with height of the value of the first component of the winning node[2] for the last movement made towards the target. For target heights of −25 cm, the value of the first component of the winning node ranges from 16 to 18; for target heights at −40 cm, the value of the first component of the winning node ranges from 8 to 11; while, for target heights at −55 cm, it ranges from 0 to 2. Thus, there is an increase in the value of the first component of winning node with height for this training run. For one servo-motion comprising four individual movements, where the z value decreases with each movement, the corresponding winning node indices for each movement are (3,19), (2,19), (1,19) and (0,19).

A further test was run on this system. In this instance, the LED representing the target to be reached was placed between the robot arm's end-effector. The system's response, i.e. its attempt to reach for the target, was either to make no movement having 'decided' that the end-effector had reached the target, or to make a small movement of the order of 1 cm. Such small movements would have been made because of the network's discrete or quantized nature, i.e. only a certain number of positions can be reached for, corresponding to the number of nodes in a well-trained network, and if the target was not exactly at such a position, a movement to the network's closest position would be made.

During the course of the work described in this paper, an exact model of how the performance varies with array size (and, attendently, the number of training iterations) was not derived. However, a rough indication can be inferred from Fig. 8, which depicts the average error in the final servo-motion position as a function of network size.

5. Conclusions

The system described in this paper is capable of reaching for a visually presented target and achieving the required position after a number of movements. The error in achieved position and the position of the target is, on average, approximately 3 cm; the smallest disparity in position achieved in tests was 1.28 cm. The system acts as a servo-mechanism, making up to four movements to reach the target when the 20×20 network is employed. Initial movements tend to be larger than later ones in a given sequence, so the system incrementally reduces the disparity between its current position and the required target position, converging on the target until the limit of the positional resolving power of the network is reached. The number of movements made in a sequence depends on the initial distance between the target and the end-effector of the robot arm. The system could, at least in principle, track a moving object although, for this implementation, not in real-time. A boundary effect was found to exist in the networks. This effect decreased with increased network size and with the number of training steps employed. In addition to this boundary effect, the system was found to have a diminished ability to reach of targets in the upper part of the robot's working envelope (or, rather, that part of the envelope which was used in training the network). This was due to the frequent inability of the cameras to image the LED when in this part of the envelope. Apart from the boundary effect and this reduced ability to reach for targets in the upper part of the envelope, the system does not display any significant non-uniformity in accuracy as a function of position in the work-space.

To place this work in the context of existing technology, the system demonstrates the feasibility of this self-organising/learning methodology. In terms of practical usefulness, however, an error of approximately 3 cm over a work-space of width 34 cm, depth 9 cm and height 35 cm compares badly with the very accurate traditional methods discussed in the introduction to this paper.

It is important to note that a wide-angle infrared LED was used as the target solely to simplify the vision processing, specifically to facilitate simple segmentation of the robot end-effector and target. Other objects could also be used as the target if more advanced vision techniques were deployed.

This work has demonstrated that increasing network size and training times produces systems which are better able to reach for the target, i.e. which make more servo movements, achieve better accuracy, and have a reduced boundary effect. With greater computational power then better systems could be produced. This is not to say that a very large network with a very large number of training steps would necessarily result in a system with arbitrarily high accuracy. Nonetheless, while the question as to the limit of attainable accuracy is not answered in this paper, it has clearly established that accuracy does increase with network size and with the number of training iterations. Unfortunately, so does the time required for training the network.

[2] The array of nodes in the Kohonen net is two-dimensional: thus, the position of every node can be described by a tuple with two components.

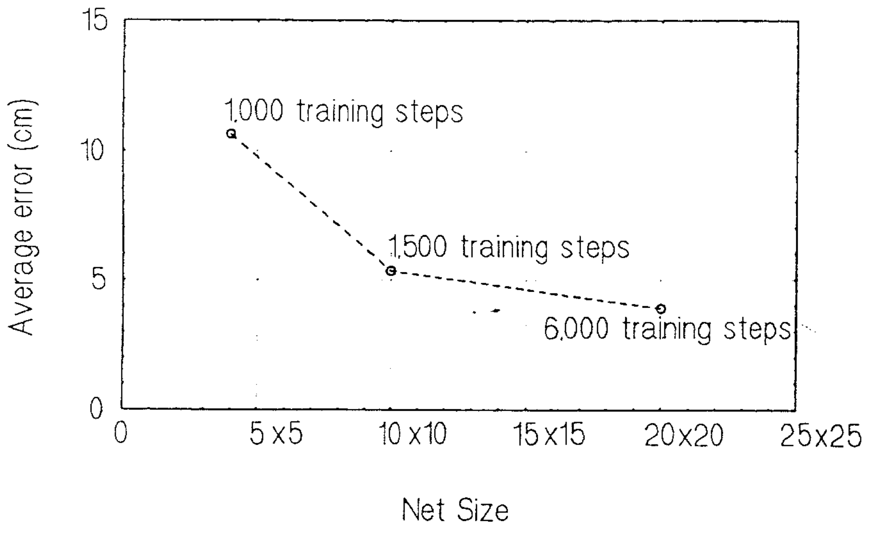

Fig. 8. Variation in the average error in final servo-motion position as a function of network size.

The merit of this system is that it is an adaptive, self-calibrating system which learns by experience, and which does not involve calculations such as those involved with camera calibration models, triangulation methods, and the development of homogeneous transformations.

Acknowledgements. The authors are grateful to Dr Yves Coiton for discussing his work during the early stages of this research. Thanks also to Dr Paul Horan for many helpful discussions.

References

1. Paul RP. Mathematics, Programming and Control. The Computer Control of Robot Manipulators. MIT Press, 1983
2. Vernon D. Machine Vision: Automated Visual Inspection and Robot Vision. Prentice-Hall, 1991
3. Snyder WE. Computer Interfacing and Control. The Computer Control of Robot Manipulators. Prentice-Hall, 1985.
4. Coiton Y, et al. A neural network model for the intersensory coordination involved in goal-directed movements. Biol Cyber 1991; 66: 167–176
5. Roll JP, et al. Proprioception as a link between body space and extra-personal space. In: J Paillard (ed.), Brain and Space, pp. 112–132, Oxford University Press, 1991
6. Held R, Hein A. Movement produced stimulation in the development of visually guided behaviour. J Comp Physiol Psychol 1963; 56: 872–876 1963
7. Jordan MI. Motor learning and the degrees of freedom problem. In: M Jeannerod (ed.), Attention and Performance xiii, pp. 796–836, Erlbaum, 1989
8. Massone L, et al. A neural network model for limb trajectory formation. Biol Cyber 1989; 61: 417–425
9. Dean J. Coding proprioceptive information to control movement to a target: Simulation with a simple neural network. Biol Cyber 1990; 63: 115–120
10. Bartlett WM. A Neurally-inspired Connectionist Approach to Learning and Performance in Vision-Based Robot Motion Planning. PhD thesis, Beckman Institute, University of Illinois, March 1989
11. Bartlett WM. Murphy: A robot that learns by doing. Am Inst Phys January 1988; 544–553
12. Kuperstein M. Neural model of adaptive hand–eye coordination for single postures. Science 1988; 1308–1311
13. Ritter HJ, et al. Topology conserving maps for learning visuo-motor coordination. Neural Net 1989; 2: 159–168
14. Martinetz M, et al. Three-dimensional neural net for learning visuomotor coordination of a robot arm. IEEE Trans Neural Networks March 1990; 1(1): 131–136
15. Ritter H, Martinetz T, Schulten K. Neural Computation and Self-Organizing Maps. Addison-Wesley, 1991
16. Walter JA, Schulten KJ. Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. IEEE Trans Neural Networks 1993; 4(1): 86–95
17. Kohonen T. Self-Organisation and Associative Memory, 2nd edn. Springer-Verlag, 1988