A NEURAL-NETWORK-BASED FLEXIBLE ASSEMBLY CONTROLLER

M D Majors and R J Richards

University of Cambridge, UK

ABSTRACT

A neural network-based learning algorithm has been developed for control of robotic peg-in-hole insertions. As simple insertions may comprise up to one-third of all assembly operations, this controller has direct application to automated flexible assembly. A neural network derived from the Cerebellar Model Articulation Controller (CMAC) utilising force feedback from a Cartesian robot is demonstrated.

INTRODUCTION

Flexible Assembly

Robots are currently under-utilised in assembly automation. Due to difficulties with precise control and slow cycle times, robots are only used in specialised applications such as printed circuit board assembly. For a robot to replace a dedicated assembly machine, it must overcome its inherent limitations and become *flexible* (i.e. be able to assemble more than one product). Only through flexibility will an assembly robot be more efficient than hard automation.

Current techniques in flexible assembly are designed to allow a robot to assemble more than one product. This is appropriate for a manufacturer whose product range comprises many different models of high-volume items. The flexible robot will be cheaper than a dedicated machine for each model and faster than manual labour.

However, economics dictate that a small producer cannot order a robot or assembly-line tailored exactly to his specifications since this is an extremely time-consuming and therefore expensive process. It would be advantageous for the factory to be able to purchase a generic assembly robot with no *a priori* knowledge of the product that must be assembled. By combining the programmability of a robot arm with the learning capabilities of a neural network , it may be possible to produce a completely generic assembly robot.

The most common task used to demonstrate the flexibility of an assembly robot controller is the basic operation of two-dimensional insertion of a peg into a hole (Figure 1). The complexity of this task involves the uncertainty of the relative positions of the peg and hole. The two-dimensional task is not unrepresentative of those found in industry since the errors of a peg in the grasp of a flat-jaw gripper will in general only be in the plane of the jaws.

Previous Work

Many possible solutions to the peg-in-hole insertion problem have been proposed, using both active and passive control methods. Passive controllers focus on the usage of a programmably compliant robot wrist. Badano et al. (3) utilise a pseudo-random motion to insert a chamferless peg with manipulator compliance. Chamfered insertion via a remote centre compliance technique is examined in Whitney (9) and refined with force feedback in Hopkins et al. (5). The latter approach is examined for chamferless pegs in Caine et al. (4).

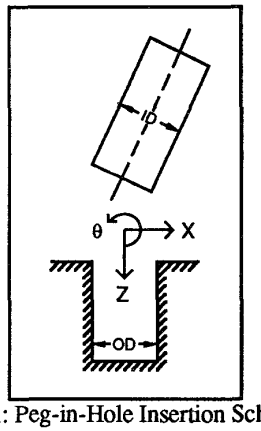

Figure 1: Peg-in-Hole Insertion Schematic

The active methods for peg-in-hole insertion rely upon feedback from force sensors in the manipulator. A binary-tree database controller has been implemented successfully by Ahn et al. (1), as has a logic branching technique by Vaaler and Seering (8). Simons et al. (7) propose a neural network-based approach although empirical data is not presented for corroboration.

CMAC BACKGROUND

An artificial neural network (ANN) is an algorithm designed to mimic the memory and learning capabilities of the human brain. The ANN consists of a set of receptive regions that are activated by certain stimuli just as neurones would be in the brain. These receptive regions correspond to a series of weights which are manipulated such that memory and generalisation are achieved. Thus the ANN can not only store correct outputs for a given input, but also interpolate/extrapolate to produce a nearly correct output for an input that is near to one it has been given before.

'Artificial Neural Networks', 26–28 June 1995, Conference Publication No. 409, © IEE, 1995

The standard terminology delineates 3 stages of the ANN learning process. The two most basic operations are *learning* and *responding*. The ANN *learns* to produce a desired output by modifying its weights such that a given input will produce the appropriate result. An ANN *responds* to a given input set by manipulating its weights to produce an output from what it has previously learned. Prior to real-time operation, the ANN is *trained* off-line. It is fed an appropriately general set of inputs along with their corresponding correct outputs and allowed many learning iterations to attempt to simulate this mapping. The ANN may be able to learn whilst in control of a system if a correct version of the output, or some measure of the correctness of the output exists at that time.

The Cerebellar Model Articulation Controller (CMAC) neural network was developed by James Albus (2) in the early 1970s from his studies of the functioning of the human brain. Because of the simplicity and corresponding speed of the CMAC algorithm, it is particularly appropriate for use in real-time control situations. As in Majors et al. (6), the CMAC architecture is described as a series of mappings between spaces (Figure 2). The most basic feature of Figure 2 is that inputs which are close in input space X, share more weights in weight space Q, which in turn produces similar outputs in output space Y.

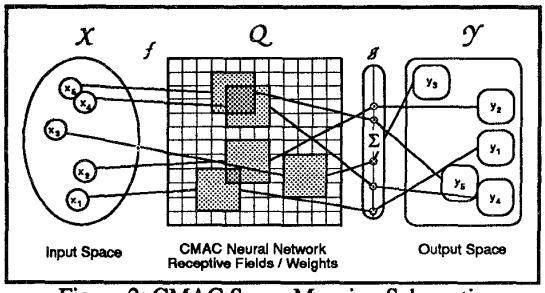

Figure 2: CMAC Space Mapping Schematic

The first operation takes the input vector $\mathbf{x}$ from input space X and maps it to the neural space Q.

$$f: X \rightarrow Q \tag{1}$$

The mapping f is the essence of the CMAC algorithm. It is performed by quantising the dimensions of the input space X each into n discrete units or receptive fields, q_j, arranged in overlapping hypercubes. Each receptive region is associated with a weight. The receptive field q_j^i is activated if the input x^i falls into the j^{th} receptive region. Each input vector will excite exactly C (the generalisation parameter) receptive fields. The function f also provides for the property of generalisation: inputs that are close in input space X will produce outputs that are close in output space Y.

The mapping f is more easily understood for a one-dimensional CMAC as shown in Figure 3.

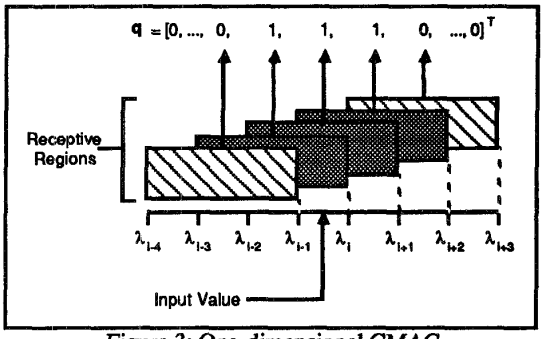

Figure 3: One-dimensional CMAC

In the one-dimensional scenario, the input space X is divided into n quantisation intervals, by n+1 divisors, λ_i. The receptive regions are arranged as overlapping line segments. Here with a generalisation parameter C equal to 3, exactly 3 receptive regions will be activated for each input. The receptive region is excited only if the input x falls into its quantisation space.

$$q_j = \begin{cases} \text{active} & \text{for } x \in [\lambda_{j-C}, \lambda_j) \\ \text{inactive} & \text{otherwise} \end{cases} \tag{2}$$

Each receptive region q_j corresponds to a weight in Q space. Generalisation is accomplished such that as an input moves one quantisation width, one receptive region becomes deactive and one new one becomes active; thus C-1 weights are shared by the neighbouring inputs and the outputs will therefore be similar.

The output of the CMAC is derived from the second mapping taking the weights $\mathbf{q}$ in Q space to the output $\mathbf{y}$ in Y space.

$$g: Q \rightarrow Y \tag{3}$$

In a simple CMAC, such as the one chosen, the function g merely sums the C active weights so that

$$g = \sum_{i=1}^{C} q_i^{active} \tag{4}$$

The training of the CMAC is accomplished through the update of the weight vector such that

$$q_{i+1} = q_i + \xi \frac{(y_d - y)}{C} \tag{5}$$

where $\mathbf{y}$ is the output of the CMAC response to a given input, $\mathbf{y}_d$ is the desired response to the same input, C is the generalisation parameter and ξ is a learning rate with a value between 0 and 1.

The mappings above describe a neural network with three key features. The generalisation parameter, C, can be controlled independently of the number of input dimensions and the size of the physical weight space. Uniform approximation is realised through the property

that a change of one quantisation interval in one input will cause exactly one receptive field to become active and one to become deactive. Finally, the receptive fields are arranged in such a fashion that summations of weights throughout the space are accomplished with a minimum of calculations.

The one-dimensional CMAC depicted in Figure 3 can easily be extended to N dimensions where C N-dimensional hypercubes are arranged in a geometrically regular pattern to achieve the desired properties. This realisation can, at its most basic, be implemented as an N-dimensional array of weights or look-up table such that the computational requirements are minimal and response is quick.

CONTROLLER DEVELOPMENT

CMAC Controller Design

A neural network based on the Cerebellar Model Articulation Controller has been chosen due to its fast internal adaptation algorithms and its generalisation capabilities. The inputs to the controller are vertical force f_z, lateral force f_x and a lateral force differential t_θ to represent moment from the gripper; the output is a movement command in either the X, θ or Z directions (Figure 1). The CMAC weight space is initialised with random movement commands. For this experiment the weight space has been pre-set with random movements but in the proper directions for one-point contact, so as to expedite the learning. The updating of the weight space is accomplished by the choice of a new random movement whenever the previous command has increased the forces.

This research is distinguished from previous work by several points. The algorithm does not rely on any *a priori* knowledge of the dynamics of the task. Secondly, the actuator hardware does not require any non-standard equipment (e.g., programmably compliant wrists or expensive load cells). Finally, the controller has been verified experimentally, rather than in simulation.

The task chosen to demonstrate the flexibility of the CMAC-based controller is the basic assembly operation of the insertion of a peg into a hole (Figure 1). The complexity of this task involves the uncertainty of the relative positions of the peg and the hole; herein lies the most immediate application to industry. The task is simplified in this experiment by the known vertical orientation of the hole. The peg is then positioned in the manipulator jaws with random errors in X and θ orientation from the ideal path for simple vertical insertion. The CMAC returns the peg to an acceptable path.

The peg-in-hole insertion task is representative of those necessary for flexible assembly in that the controller has very little *a priori* knowledge of the dynamics of the system to be controlled; it will learn to perform the task based simply upon a completion rule. Furthermore, the controller will be subjected to slight variations in operating conditions for which it must learn to compensate.

The CMAC-based controller is not seeking to determine an optimal path from initial to final conditions. Instead, an acceptable solution is sought through trial-and-error learning. Speed of learning rather than speed of operation is the impetus for the search.

CMAC Controller Algorithm

As the feedback from the strain gauges provides force measurements in three directions (f_z,f_x,t_θ), a three-dimensional CMAC was chosen for the controller. In this configuration, the layers of receptive fields are organised as three-dimensional hypercubes.

Each receptive field contains the magnitude of a movement in either or both the X and θ directions. Thus the CMAC is essentially an associative device between a set of forces between peg and hole and the appropriate corrective movements for vertical insertion.

It is assumed that to return to the peg to an acceptable path for insertion that the forces resulting from the process should be minimised. The algorithm for the CMAC to perform the peg-in-hole insertion is shown in Figure 4.

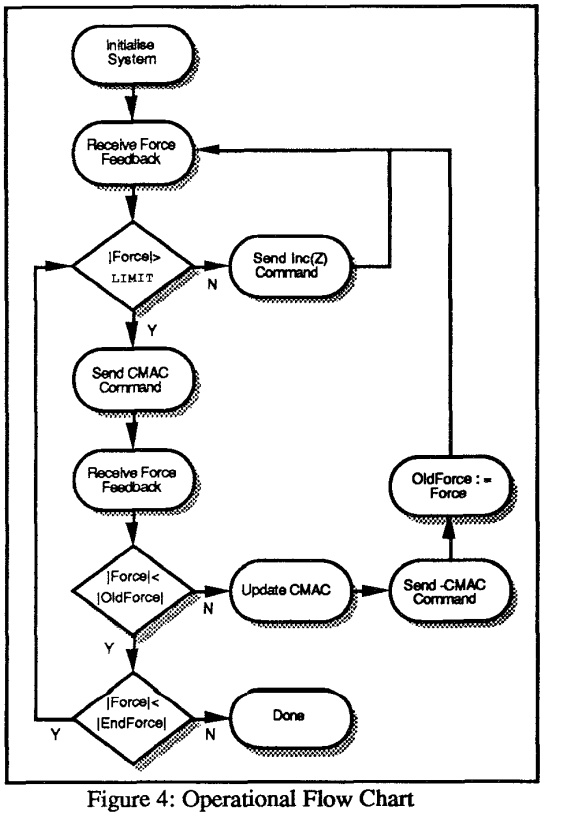

Figure 4: Operational Flow Chart

The initialisation of the system introduces artificial random errors in the X and θ positions of the peg above the hole to simulate inaccuracies in robotic placement. The

forces on the gripper are then determined and the Cartesian norm of

$$F = \sqrt{f_z^2 + f_x^2 + t_\theta^2} \qquad (6)$$

is obtained. If the aggregate force, F, is below a LIMIT then the robot is sent a command to move a small increment in the Z direction. If the force is greater than the LIMIT, then the CMAC response to the force set is determined and appropriate corrective motions in the X and θ directions are commanded to the robot.

If the aggregate force, F, has increased over the previous movement (indicating a positive gradient) then that movement is considered to have not contributed to the likelihood of a successful insertion. Therefore, the movement is retracted. The CMAC learns from its incorrect experimentation, choosing new random movements (direction and magnitude) for those areas of the weight space that had contained erroneous information. The process repeats with the measurement of forces until the end condition is met.

CMAC Controller Implementation

The test robot used was an IBM 7565 (Figure 5). The IBM 7565 is a Cartesian-type robot with 6 degrees-of-freedom motion capabilities. Each gripper finger is equipped with strain gauges able to measure forces in the X, Y, and Z directions.

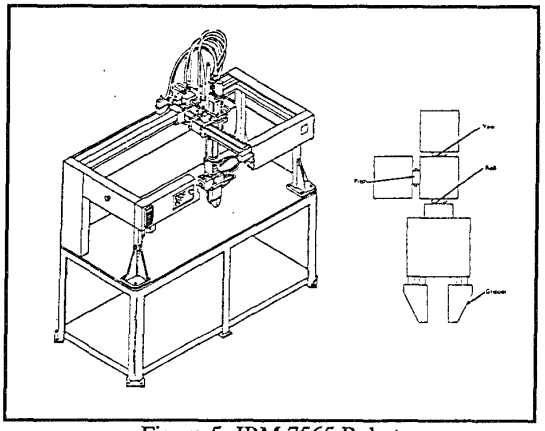

Figure 5: IBM 7565 Robot

The IBM 7565 is controlled by a dedicated processor, the IBM Series I controller. To decrease the dependence of the system upon the robot controller, the CMAC is coded in a 486 PC that relays its movement commands along a serial line to the Series I. This allows the controller to remain relatively independent of both robot dynamics and obscure robot programming languages.

A soda can was used as the peg for this preliminary stage of testing. The can was filled with an expanding foam to provide some measure of durability while the aluminium skin allowed for error-driven crushing which might otherwise have led to robot damage. Two vertical plates served as the hole. The soda can has a diameter (ID) of 65.6mm and the hole has a variable width (OD) providing a range of clearances:

$$Clr = \frac{OD - ID}{OD} \qquad (7)$$

The parameters of the CMAC network were chosen both heuristically and arbitrarily. The upper and lower boundaries on the strain gauge outputs were found through several trial insertions to be ±1500 (units). The number of quantisation intervals per input dimension and the number of layers were each chosen at random (equal to 10) to eliminate the dependence of the results upon these values.

The value of the LIMIT force beneath which vertical insertion can take place should ideally only be related to frictional forces between the peg and hole. However, the force chosen appeared to be not only independent of friction, but to influence results highly. A LIMIT of 160 was chosen to eliminate anomalous force readings whilst the peg was not in contact with the hole due to inertia.

EXPERIMENTAL RESULTS

The applicability of the chosen algorithm to the intended task is demonstrated through two testing scenarios. First the clearance of the peg-hole system is varied while the initial error positions of the peg are held constant. Secondly, the generalisation and learning capabilities of the CMAC are illustrated through training with successively more difficult conditions. In each of the following figures, the upper set of graphs represents the total number of CMAC-controlled moves necessary for successful insertion at each trial. The lower graphs show the number of moves that were retracted (and also adapted) during the course of each successful insertion.

Figure 6 confirms that the difficulty of the insertion is directly dependent upon the clearance between the peg and the hole. With a clearance of 13%, the initial weight set is sufficient to perform the insertion without any further adaptation. The operation requires between 10 and 14 moves. With the clearance decreased to 10%, the insertion initially required 24 moves, of which 2 were incorrect and were updated. After 5 insertions, the number of adapted moves decreased to zero and indicated that convergence of the weight-space had been achieved. At this point, the insertion required between 16 and 18 moves for success. At 7% clearance, the first insertion needed 62 moves and 15 adaptive moves. At the second trial, the number of moves increased to 80. However the number of incorrect moves decreased accordingly to 9. After 7 trials, the weights converge and the insertion required between 30 and 35 moves.

Despite the steady decrease in the number of incorrect and adapted moves, the number of moves necessary for a successful insertion is not monotonically decreasing. This apparent discrepancy is due to three factors. First the system is extremely noisy such that even after com-

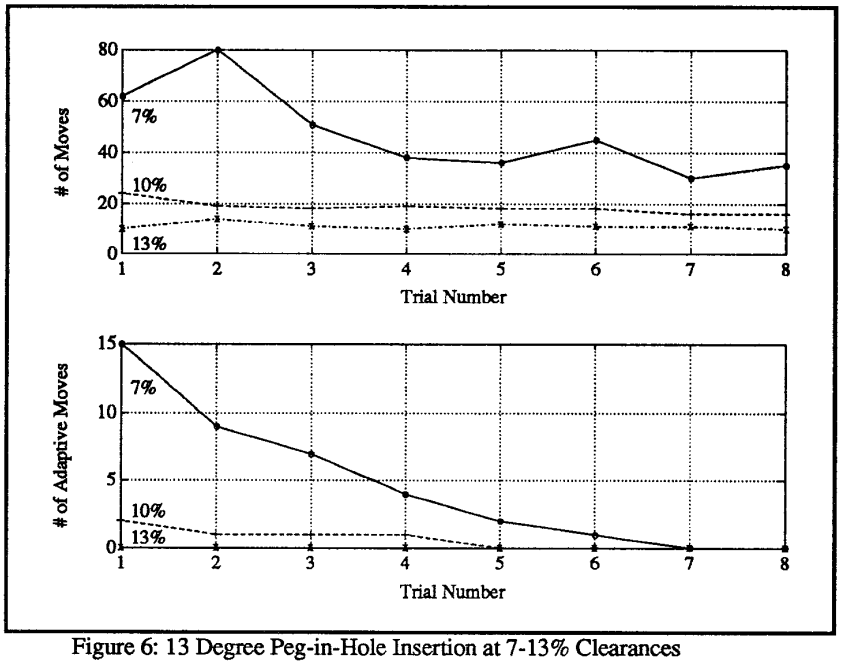

Figure 6: 13 Degree Peg-in-Hole Insertion at 7-13% Clearances

plete convergence the number of moves is not constant. Secondly, when the controller retracts an incorrect move, it does not return the system to exactly the same state that it was in before the incorrect move was performed. Therefore a non-completely determined path has been followed. Finally, the non-optimal nature of the random searching algorithm results in a newly chosen move occasionally worse than the original.

To demonstrate that the training of the controller on one scenario has applicability to others, the CMAC is allowed to learn to perform insertions from an initial error condition of 12° and then the task is increased in difficulty to 13°. The results of this testing at 7% clearance are shown in Figure 7. It is evident that after the first trial, the previously trained controller consistently required less moves and converged more rapidly

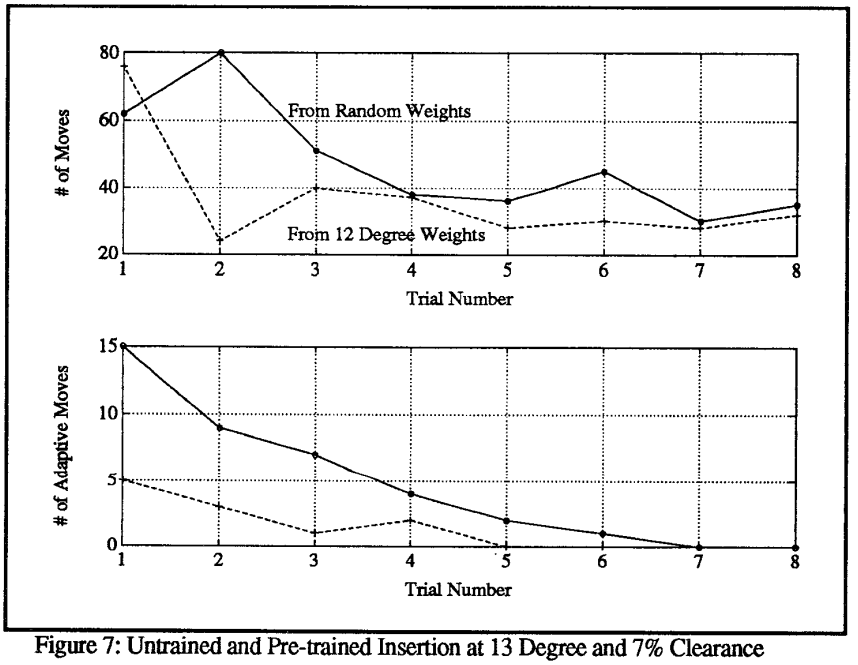

Figure 7: Untrained and Pre-trained Insertion at 13 Degree and 7% Clearance

CONCLUSIONS

Figure 7 shows that the CMAC-based controller is flexible in the conventional sense. It is be able to adapt to slight variety in the insertion task. Figure 6 shows the more general flexibility of the controller: It is clearly able to learn to perform the desired task with very little *a priori* information. It appears that the CMAC assembly controller has the potential to provide the basis for a flexible robotic system.

REFERENCES

1. Ahn, DS, Cho, HS, Ide, K, Miyazaki, F, Arimoto, S, 1992, "Learning Task Strategies in Robotic Assembly Systems", Robotica, 10, 409-418

2. Albus, J, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)", 1975, ASME Trans.: J. Dyn. Sys., Meas. and Cont., 220-227

3. Badano, F, Betemps, M, Redarce, T, Jutard, A, "Robotic Assembly by Slight Random Movements", 1991, Robotica, 9, 23-29

4. Caine, ME, Lozano-Perez, T, Seering, WP, "Assembly Strategies for Chamferless Parts", 1989, Proc. IEEE Int. Conf. Rob. Auto., 472-477

5. Hopkins, SH, Bland, CJ, Wu, MH, "Force Sensing as an Aid to Assembly", 1991, Int. J. Prod. Res., 29, 293-301

6. Majors, M, Stori, J, Cho, D, "Neural Network Control of Automotive Fuel-Injection Systems", 1994, IEEE Cont. Sys., 14, 31-36

7. Simons, J, Van Brussel, H, De Schutter, I, Verhaert, J, "A Self-Learning Automaton with Variable Resolution for High Precision Assembly by Industrial Robots", 1982, IEEE Trans. Auto. Cont., AC-27, 1109-1113

8. Vaaler, E, Seering, W, "Automated Assembly with Systems Having Significant Manipulator and Part Location Errors", 1987, Proc. IEEE Int. Conf. on Robot. and Auto., 1357-1360

9. Whitney, DE, "Quasi-Static Assembly of Compliantly Supported Rigid Parts", 1981, ASME Trans.: J. Dyn. Sys., Meas. and Cont., 104, 65-77

ACKNOWLEDGEMENTS

This material is based upon work completed whilst M D Majors was supported under a National Science Foundation Graduate Research Fellowship.

Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.