

**Self Organizing Feature Maps
and Their Application To Robotics**

**MS-CIS-91-46
GRASP LAB 268**

Craig Sayers

**Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389**

May 1991

Self Organizing Feature Maps and their Application to Robotics

Craig Sayers

April, 1991

Submitted in partial fulfillment of the requirements for the Written Preliminary Exam Part II.

© 1991, The University of Pennsylvania, Philadelphia PA 19104, U.S.A.

This work was supported in part by the National Science Foundation under Grant Number BCS-89-01352. Any opinions, findings, conclusions or recommendations expressed in this report are those of the author and do not necessarily reflect the views of the National Science Foundation.

Summary

The self-organizing feature maps developed by Kohonen appear to capture some of the advantages of the natural systems on which they are based. A summary of the operation of this form of artificial neural network is presented. It was concluded that the primary benefits of using self-organizing feature maps result from their adaptability and plasticity while most problems are largely caused by the lack of a rigorous mathematical foundation.

Two different robotics applications are described. In the first, developed by Martinez and Schulten, a hierarchical structure composed of many self-organizing feature maps is used to control a five degree of freedom robot arm. While it was noted that there may be some practical problems, the general idea of using a hierarchical structure appears sound and may be applicable to a wider range of problems.

The second robotics application was developed by Saxon and Mukerjee. They used a single self-organizing feature map to learn the motion map of a two degree of freedom arm. The use of such a system should simplify path planning by combining multiple constraints into a 2-D structure.

Contents

1. Introduction.....	1
2. Biological Neurons.....	3
2.1 Natural Neurons	3
2.2 Natural Feature Maps	3
3. Kohonen's Self-Organizing Feature Maps	5
3.1 The Artificial Neuron.....	5
3.2 The Learning Algorithm.....	6
3.3 Example Applications.....	6
3.4 Discussion.....	7
3.4.1 Selecting Appropriate Parameters	7
3.4.2 Advantages	13
3.4.3 Disadvantages	13
3.4.4 The Use of a Simplified Algorithm.....	14
4. Hierarchical Neural Net for Learning Control of a Robot's Arm and Gripper.....	15
4.1 System Operation	15
4.2 Discussion.....	17
4.2.1 Comparison with Kohonen's Work.....	17
4.2.2 Practical Problems	17
4.2.3 Wider applications	18
5. Learning the Motion Map of a Robot Arm	19
5.1 System Operation	19
5.3 Discussion.....	22
5.3.1 Comparison with Kohonen's Work.....	22
5.3.2 Examination of Results.....	22
5.3.3 Alternative Implementations	23
6. Conclusions	31
7. References.....	33

1. Introduction

The aim of this report is to present a critical review of four papers. The first two papers [1,2] serve to provide an introduction to Kohonen's self-organizing feature maps; while the latter two [3,4] describe robotic applications for this type of artificial neural network.

Natural neural networks (described briefly in Section 2) seem to handle some tasks very well and in many cases are superior to existing artificial alternatives. This is particularly true in the field of robotics. For example there are few, if any, robot controllers which could manage the task of picking up this report and turning its pages. There would thus appear to be some justification in trying to mimic the actions of biological networks in an attempt to duplicate their abilities.

The self-organizing feature maps developed by Kohonen (see Section 3) are an attempt to mimic the apparent actions of a small class of biological neural networks. The idea is to create an artificial network which can learn, without supervision, an abstract representation of some sensory input.

In the robotics application described by Martinez and Schulten (see Section 4) a hierarchical structure consisting of a number of self-organizing feature maps is used to convert the signals from two video cameras into an abstract representation of the position and orientation of a cylindrical object. That representation is then used to control the motion of a robot as it moves to grasp the object.

Saxon and Mukerjee (see Section 5) describe a different application in which a single self-organizing feature map is used to create an abstract representation of both the joint and cartesian positions for a simple manipulator. This representation can then be used for collision-free trajectory planning.

2. Biological Neurons

A detailed description of biological neural networks is beyond the scope of this report. However, in order to evaluate the validity of attempting to mimic natural networks it is necessary to have some knowledge of their structure.

2.1 Natural Neurons

Each natural neuron consists of a series of input channels termed dendrites. Input signals pass from the dendrites to the cell body where a "decision" is made as to whether or not to produce an output pulse. These pulses are sent from the neuron via an output channel (axon) which connects to the input of other neurons via input/output connections (synapses) [5].

The frequency with which output pulses are generated is a function of the input signals. If very strong excitatory inputs are applied then a high frequency pulse train will result. Alternatively if a strong inhibitory input is applied then the output frequency will be reduced.

It is important to realize that while at any one time the neuron is essentially only making a binary decision (whether or not to produce an output pulse) the cumulative effect of many such decisions over time is to produce a pulse-modulated output signal whose frequency encodes the output strength. Each neuron therefore produces an analogue output [6] which may be considered a weighted sum of its input frequencies [7].

2.2 Natural Feature Maps

It seems clear that, at least as far as mammals are concerned, the brain does contain some record of previous sensory input. Furthermore, since the number of possible inputs far exceeds the number of neurons in the brain it seems clear that some form of compression (or abstraction) is taking place.

The mechanism by which the brain stores these compressed representations is not well understood however it is hypothesized [8] that they are recorded in "2-D metrically and topologically organized" sheets of neural cells. In other words the brain forms a mapping between some multi-dimensional input space and a 2-D set of neurons.

There is some evidence to support this idea. For example a set of cells have been found in the auditory cortex of mammals which respond selectively to sounds of different frequencies such that the location of a responding neuron appears to be directly related to the frequency of the input sound [9]. Other topographic representations have been found in the visual, somatosensory and motor-cortex areas of the brain [10].

Kohonen's net is an attempt to model, albeit in a very simplified form, the actions of this type of network.

3. Kohonen's Self-Organizing Feature Maps

The basic idea behind Kohonen's network [11,12] is the formation of a 2-D array of interconnected neurons. When an input pattern is presented to the network the response of each neuron is evaluated and the one which produced the maximum response, as well as those adjacent to it in the array, are modified so as to produce a stronger response to that input.

After a number of presentations of each input pattern the system should ideally reach a state where an "ordered image" of the input is stored in the network.

3.1 The Artificial Neuron

In his earlier work Kohonen's artificial neurons were of the form:

$$n_i = \mathbf{w}_i \cdot \mathbf{x}$$

Where n_i is the output from the i 'th neuron, \mathbf{w}_i is the weight vector for the i 'th neuron and \mathbf{x} is the input applied to every neuron.

Each neuron's output is thus a weighted sum of its inputs - this formulation being derived directly from models of natural neurons (see Section 2.1).

The neuron "learns" by modifying its weight vector so that it is more closely aligned with an input. If it is assumed that both the input and weight vectors are approximately normalized then the output of the neuron is an approximate measure of the cosine of the angle between its weight vector and the input. Thus a strong response would indicate a close match between the two while a weak response would indicate some disparity between the weight and input vectors [13].

In later work this neural model was further simplified by computing the output of each neuron as:

$$n_i = || \mathbf{w}_i - \mathbf{x} ||$$

This formulation removes the necessity of providing any form of normalization and the output is now a direct measure of the euclidian distance between the weight and input vectors.

The neuron "learns" to respond more strongly to an input by modifying its weight vector according to the formula:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + a(t) [\mathbf{x}(t) - \mathbf{w}_i(t)]$$

This in effect moves the weight vector closer to the input vector by an amount determined by the gain factor $a(t)$.

3.2 The Learning Algorithm

The complete learning algorithm is:

- i) Apply an input pattern
- ii) Compute the response of each neuron as $\| \mathbf{x}(t) - \mathbf{w}_i(t) \|^2$
- iii) Find the neuron with the best (minimum) response. Call that neuron c .
- iv) Update the weight vectors for the neuron c and for neurons within some neighborhood of c in the array according to the formula:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + a(t) [\mathbf{x}(t) - \mathbf{w}_i(t)]$$

By updating the weights of neurons in a neighborhood around the maximally responsive neuron the system ensures that similar inputs will illicit a response in nearby neurons while dissimilar inputs will produce a response from neurons further away in the network. The system thereby tends to adopt an ordered state.

3.3 Example Applications

Kohonen presents two examples which illustrate the use of self-organizing feature maps.

The first is in the field of speech recognition. The speech input is sampled, a fourier transform is performed and the resulting frequency components (represented by 15 analogue signals) are then fed as input to a 2-D self-organizing network.

The results after much training appear to indicate that the system has learned to respond to the input in an ordered fashion.

This is a particularly impressive example because the system has created an abstract representation of a 15-dimensional input using only two dimensions

while apparently maintaining sufficient information to permit accurate speech recognition [14,15].

The second example uses two robot arms. In this case training was accomplished by selecting a point at random in a rectangular region in the common workspace of both robots and then moving the end-effectors such that they were coincident with that point. The four joint angles (two from each robot) were then fed as input to the network. After many such trials the system learned an ordered mapping over the input space.

Once this mapping has been learned the system can be used to perform inverse transforms since the weights at each position in the array have directly encoded the joint-space positions for both robots.

3.4 Discussion

3.4.1 Selecting Appropriate Parameters

The selection of a suitable gain factor, $a(t)$, appears to be a compromise between learning rate and learning accuracy. If a high gain factor is chosen then learning will proceed rapidly but the net will be constantly shifting with each new input pattern and so the entire input space may not be evenly categorized. Alternatively, if a low gain factor is chosen then learning will be very slow but should eventually reach a very accurate result. The gain factor is therefore typically chosen to be initially rather large (to ensure rapid initial learning) and is then reduced to increase the accuracy of the final result.

By way of an example consider a 2-D array of neurons which is required to learn two input signals. Both the inputs vary between 0 and 1 with a flat probability distribution (see Figure 3.1). The response of the network is shown in Figure 3.2. Each point in the graph represents the point in the input space to which one neuron has become maximally responsive. The lines joining these points indicate nearest-neighbor connections between neurons in the square array. Initially all the weights were assigned small random values.

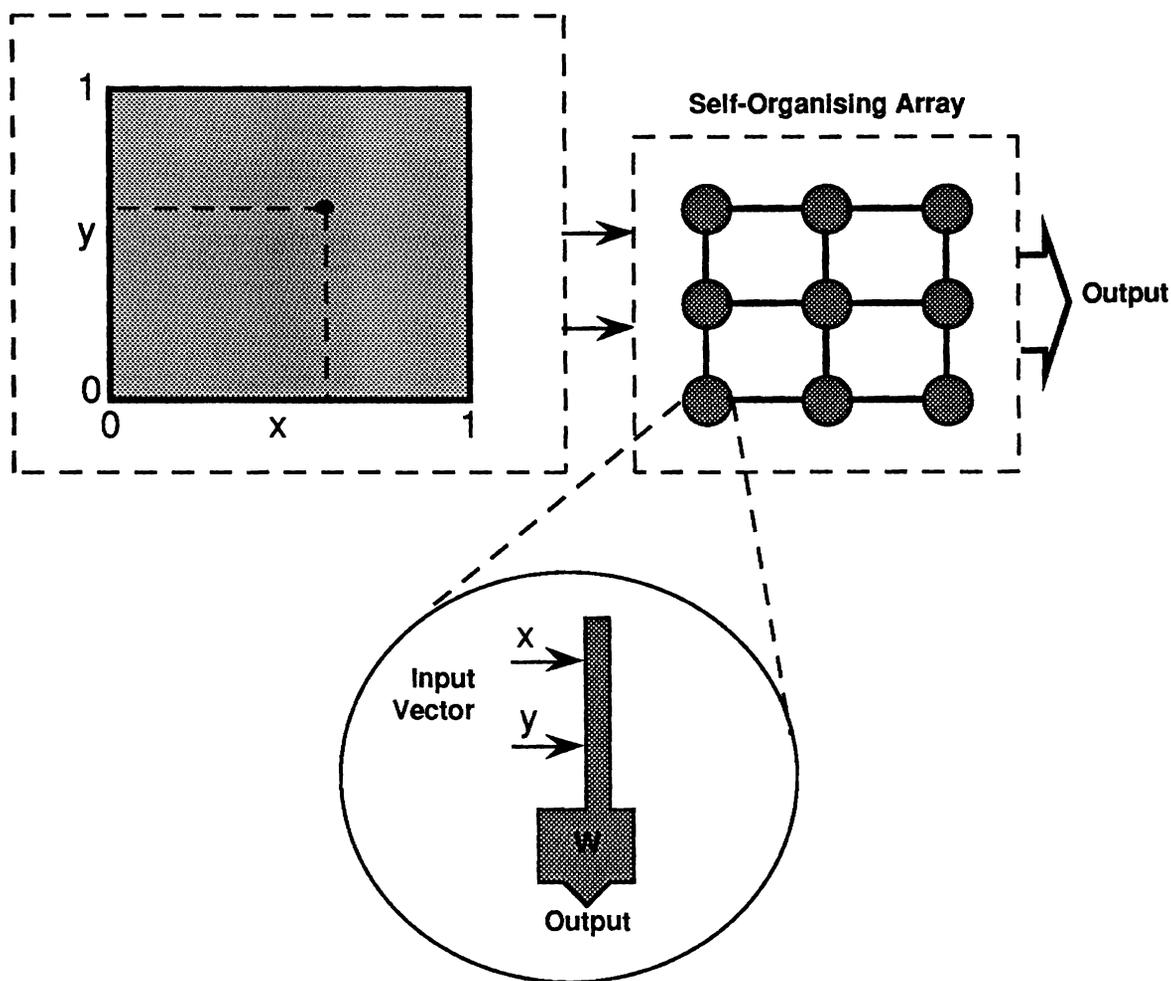
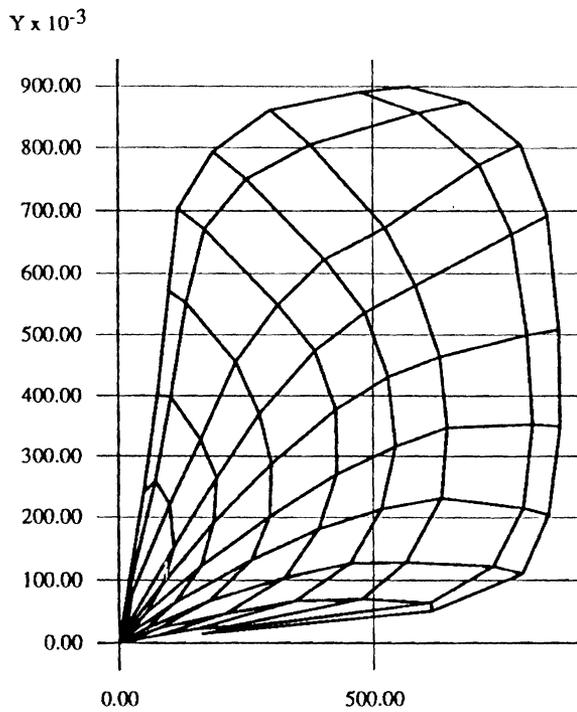


Figure 3.1 Example application in which each neuron receives two inputs.

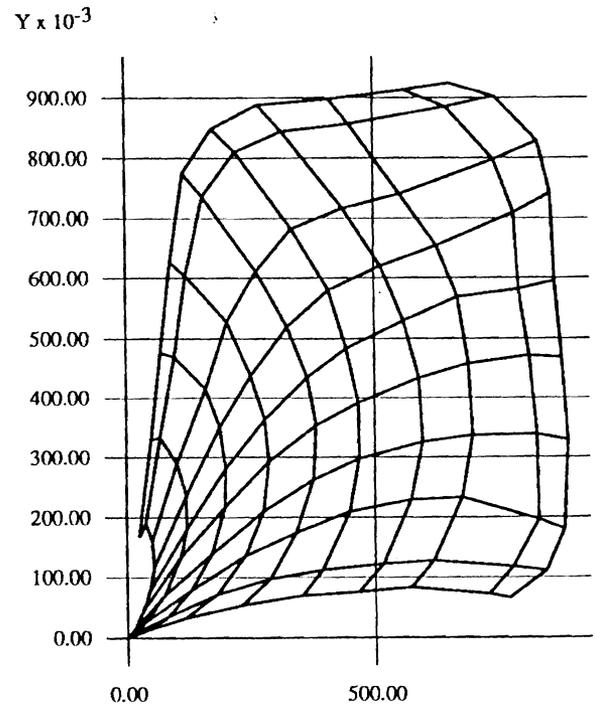
As the learning progressed the points to which each neuron responds have moved from their initial position very close to the origin and have expanded outwards to form a more even distribution over the input space. If learning were permitted to continue the network would eventually form a near-perfect square grid over the input space.

In this case the gain factor, $a(t)$, was chosen to be the constant 0.01 and it is clear that while the learning is proceeding very smoothly it is also very slow.

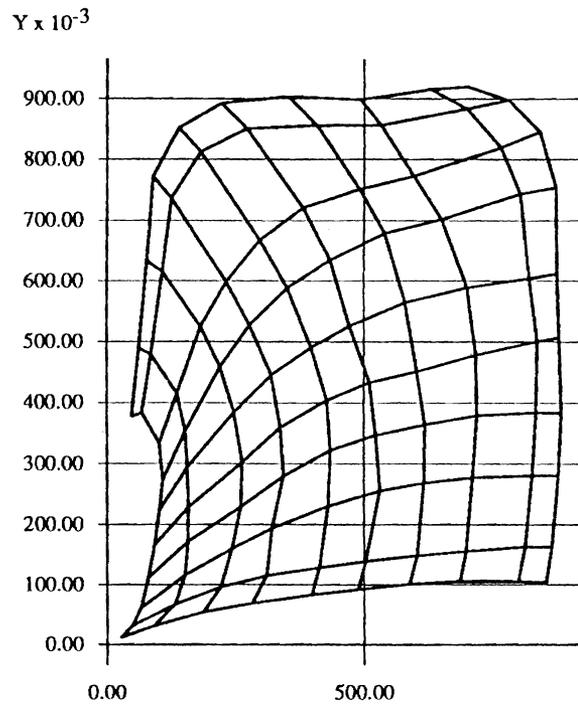
a) Response after 10000 trials



b) Response after 20000 trials



c) Response after 30000 trials



d) Response after 40000 trials

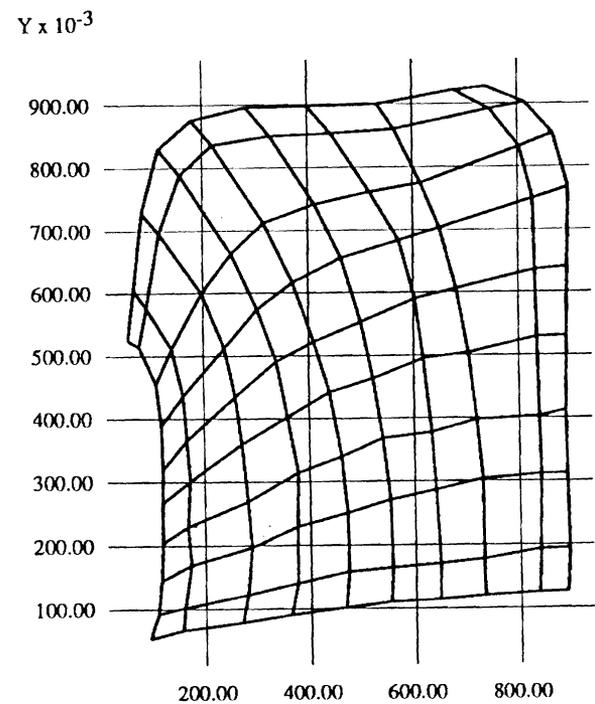


Figure 3.2 The response of a 2-D network as it learns a topological mapping over a 2-D input space. In this case the gain factor, $a(t)$, was 0.01

The effect of choosing a larger constant value for $a(t)$ is shown in Figure 3.3 where a value of 0.1 was used. In this case learning is much more rapid, reaching a square grid after less than 20000 input presentations. However, the grid is not very evenly distributed over the input space.

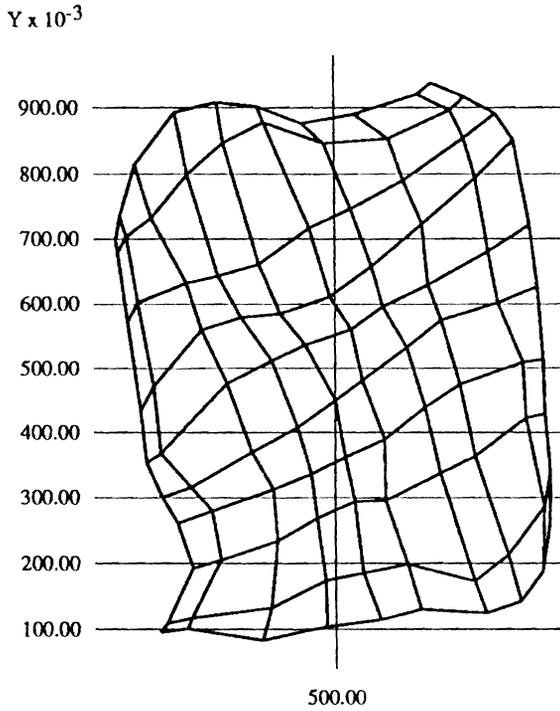
Figure 3.4 demonstrates the effect of having a time-varying learning rate. In this case an initial rate of 0.1 was used for the first 20000 input patterns and this was followed by a further 20000 presentations with the learning rate set at 0.01. It should be evident that this produces a better result than either of the two previous cases.

As the above examples demonstrate there is a considerable difference between the effectiveness of the network when different gain factor expressions are utilized. In the example given here it was known a priori that the result should be a square grid and so it was relatively easy to determine, via experimentation, suitable values for $a(t)$. However, if the dimensionality of the input were to be much larger (for example in the case of speech recognition) then it would not be such a trivial matter to determine suitable gains.

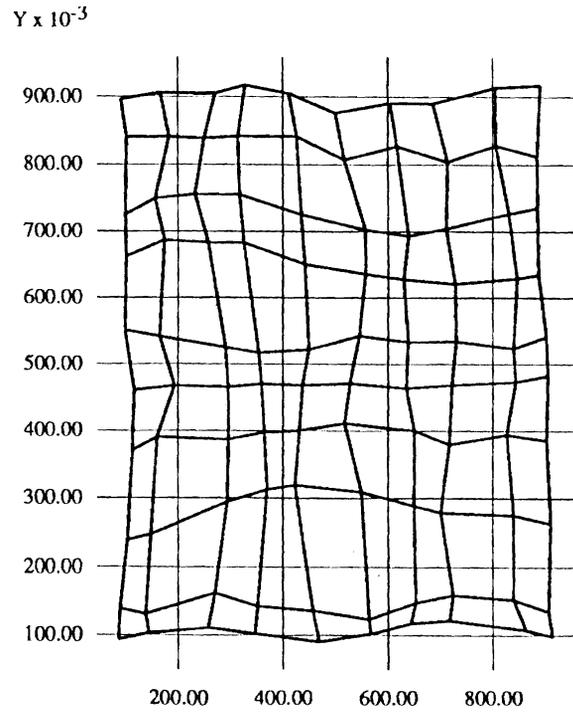
Kohonen has proven [16] that a one-dimensional network of neurons with a single input will form an ordered mapping provided that $0 < a(t) < 1$. However it is not clear that this proof would scale for cases where either the input or the network were of higher dimensionality. In particular it is unlikely that it would apply for the case where the input was of much higher dimension than the network.

The decision as to what size of neighborhood is to be adjusted also appears to be determined in a somewhat ad hoc manor [17].

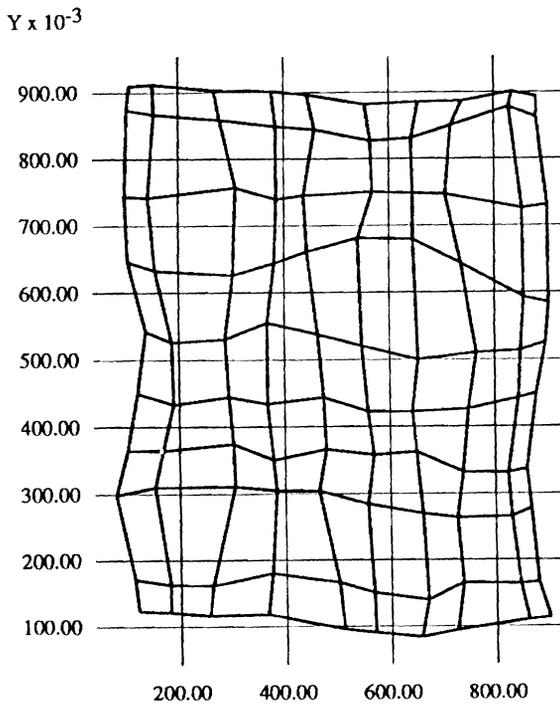
a) Response after 10000 trials



b) Response after 20000 trials



c) Response after 30000 trials



d) Response after 40000 trials

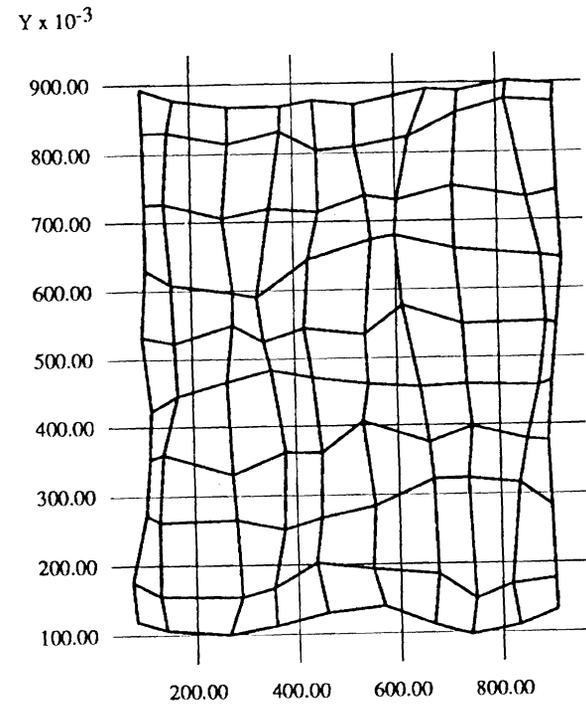
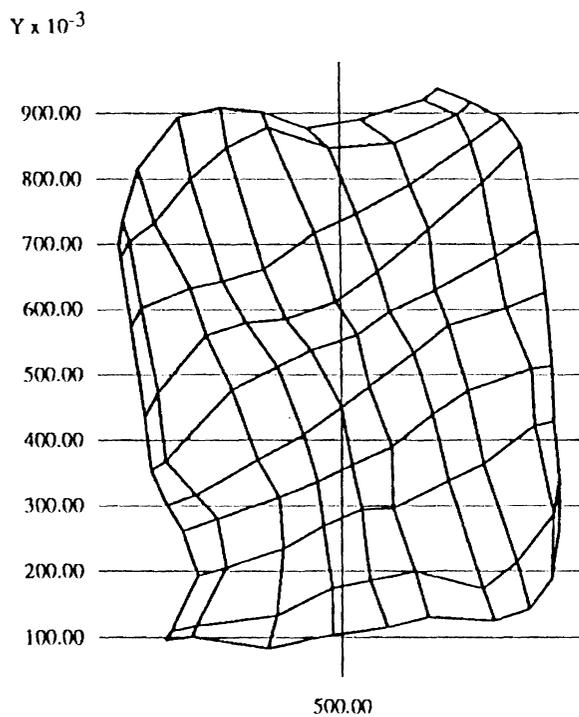
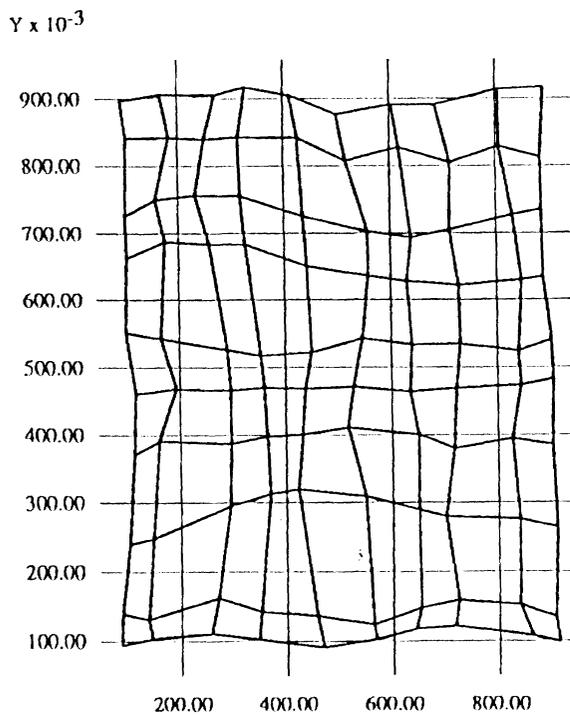


Figure 3.3 The response of a 2-D network as it learns a topological mapping over a 2-D input space. In this case the gain factor, $a(t)$, was 0.1

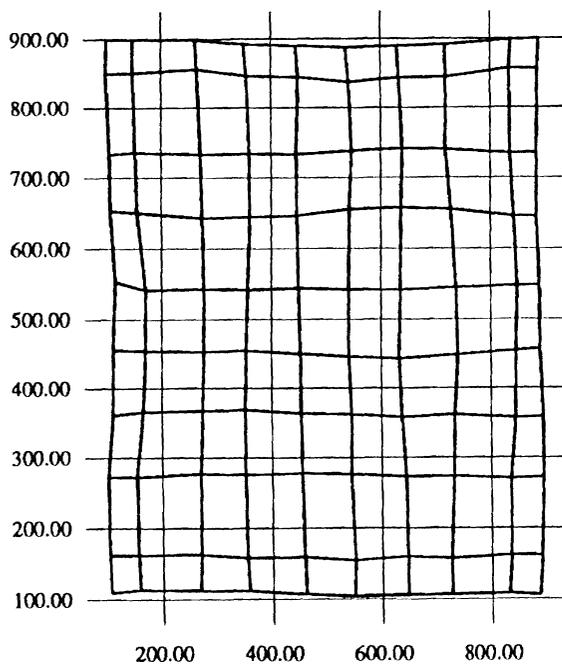
a) Response after 10000 trials



b) Response after 20000 trials



c) Response after 30000 trials



d) Response after 40000 trials

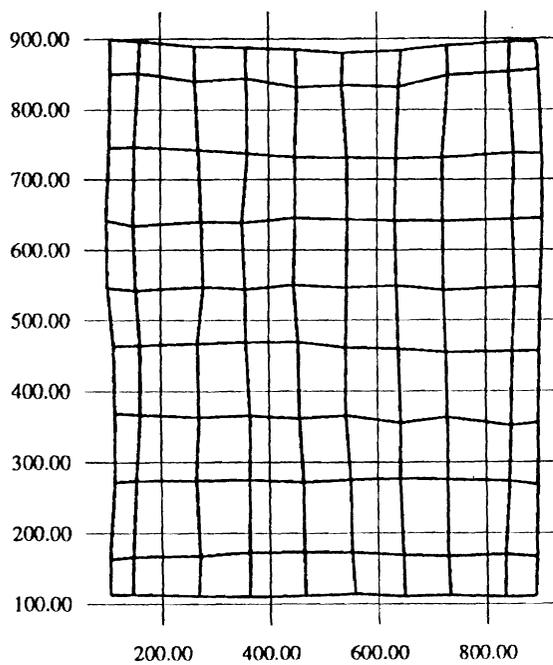


Figure 3.4 The response of a 2-D network as it learns a topological mapping over a 2-D input space. In this case the gain factor, $a(t)$, was 0.1 for the first 20000 input presentations and 0.01 thereafter.

3.4.2 Advantages

At first glance the advantages of using a self-organizing feature map over the use of simpler techniques (such as look-up tables) are not apparent and it is only when one considers the learning process that they become evident.

The primary benefit is that the network learns autonomously without the requirement that the system be well defined. It is therefore possible to learn abstract representations of systems (for example speech recognition) where the relationship between the inputs is not known.

A second advantage is that the system does not stop learning but instead continues to adapt to changing inputs. This plasticity allows it to adapt as the environment changes (see Section 5.3.3 for examples).

A particular advantage over other artificial neural networks is that the system appears well suited to parallel computation. Indeed the only global knowledge required by each neuron is the current input to the network and the position within the array of the neuron which produced the maximum output. With these two pieces of information each neuron can respond and adapt to changing inputs without the need for any other inter-neural communication. Given the suitability of this form of network to parallel processing it is surprising that Kohonen makes so little mention of this.

3.4.3 Disadvantages

One of the major problems is that for systems which are well defined it will often be much faster to calculate an exact answer rather than using a self-organizing map to obtain an approximate solution. This is particularly true when a sequential processor is employed.

Perhaps a more serious problem relates to the fact, mentioned earlier, that the behavior of such networks is not well defined mathematically. This means that the selection of network parameters (such as gain factors and neighborhood sizes) must be performed in a somewhat ad hoc manor. Of particular concern is apparent lack of any guarantee that a network, allowed to adapt in the real world, will always maintain a consistent ordered mapping.

3.4.4 The Use of a Simplified Algorithm

Kohonen appears to give the impression [18-20] that the learning algorithm which he implements (see Section 3.2) is a simplified version of a more biologically-correct model. However, if this were the case then one might expect that the self-organizing behavior which the simplified version exhibits would also be present in the more complex system. Recent work by Acker and Kurtz [21] has shown that this is not the case. It would appear that the self-organizing behavior of Kohonen's simplified algorithm is only a property of that particular implementation. While this doesn't necessarily mean that the simplified form is incorrect it does suggest that attempts to justify such a formulation on a biological basis should be treated with some skepticism.

4. Hierarchical Neural Net for Learning Control of a Robot's Arm and Gripper

The network developed by Martinez and Schulten [22] uses a hierarchical structure to learn a mapping between the spatial position of a cylindrical object and the joint-space position which a five degree of freedom robot must adopt in order to grasp that object.

4.1 System Operation

The inputs to the highest level of the system are four numbers representing the (x,y) position of an object in the visual field of two video cameras. This input is passed to a 3-D array of neurons which learns, using Kohonen's method, a topological mapping between the 4-D input and the 3-D array. The result is that each neuron comes to represent a particular spatial position for the object.

Associated with each of these neurons in the 3-D net is an additional 2-D array of neurons. These arrays receive as input the orientation of the cylinder as seen by the two video cameras and they learn a mapping between these inputs and the 2-D array.

The classification of the input picture into both a position and orientation is therefore a two-step process. Firstly the 3-D net is used to recognize the position of the cylinder and then the 2-D net associated with that position is used to determine the orientation of the cylinder (see Figure 4.1). It should be noted that the output from this process is simply the position within each network of the maximally responsive neuron - the output is not in the form of a cartesian position/orientation.

Each neuron in the 3-D net has associated with it the particular posture which the arm must attain in order to grasp a cylinder at the spatial position encoded by that neuron. Stored along with this posture is a jacobian matrix representation of the relationship between changes in the positional information from the cameras and changes in the arm posture.

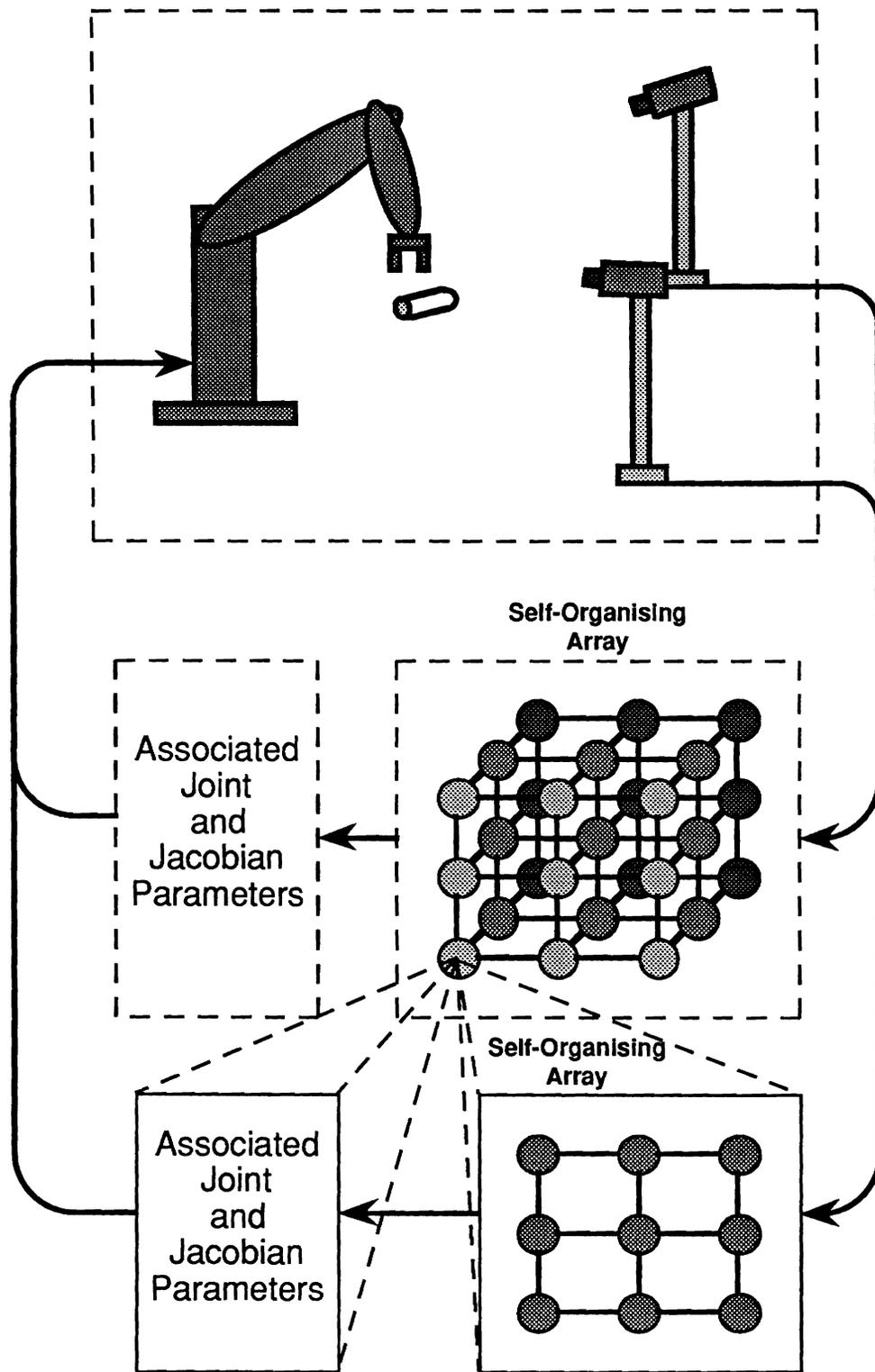


Figure 4.1 The hierarchical system employed by Martinez and Schulten.

Each neuron in each of the 2-D nets has associated with it the particular wrist configuration which is required in order to grasp the cylinder when it is orientated at the angle encoded by that neuron. Each neuron also stores a jacobian representing the relation between changes in the orientational information from the cameras and changes in the wrist orientation.

The system is able to place an arm in approximately the correct position by using the stored joint positions. It can then compare the current end-effector position (as recorded by the two video cameras) with the desired end-effector position and make use of the jacobian matrices associated with the current arm position to reduce this error.

4.2 Discussion

4.2.1 Comparison with Kohonen's Work

The use of a 3-D network represents a departure from the 2-D style which Kohonen employed. However, Kohonen's decision to only utilize one and two-dimensional structures appears to have been motivated by the assumption that natural networks only employ these forms and this need not necessarily be the case [23].

4.2.2 Practical Problems

The error correction between the current and desired end-effector positions is performed in terms of differences between the camera views of the cylinder to be grasped and the camera views of an imaginary cylinder in the jaws of the gripper. The method by which the system can determine the appropriate positional data for this imaginary cylinder based only upon two views of the end-effector is not apparent.

Furthermore the authors appear not to have considered the possibility that the cylinder to be grasped may block at least one camera's view of the end-effector.

Another possible problem is that no allowance is made for the space occupied by the cylinder during positioning. It is almost certain that the arm would attempt to move through the cylinder while attempting to grasp it.

One way to overcome these difficulties may be to incorporate an additional network which could relate a joint-space position to an image of the arm. Once this network had been trained the system could use it to "imagine" where the arm would be after each motion. This would allow the joint parameters to be corrected without the necessity of actually moving the arm.

A more serious problem is that the current system only associates a single joint-space position with each location for the cylinder. This ignores the possibility that redundant configurations may exist. A similar system has been implemented using a redundant arm [24], however that implementation was still restricted to associating a single joint-space position with each cartesian point - thereby negating many of the advantages of using a redundant manipulator.

4.2.3 Wider applications

In the application described by Martinez et al the use of a hierarchical network was influenced by the structure of the robot arm and the lower layer consisted of multiple 2-D arrays. However it is possible that an alternative hierarchical structure may also be feasible.

If only a single network is used then the network size is a compromise between having a small network which requires little computation but gives a rather coarse result and a large network which requires much computation but produces a very accurate result. The necessity of making this compromise could be eliminated by employing a hierarchical structure in which a small network was used to select a small subsection of a much larger network. This would only require the evaluation of two small networks (the smaller net and a subsection of the larger one) yet it may give a similar accuracy to that achieved using the whole of the larger network.

It would also appear to provide an excellent way of evaluating a large network using a small array of processors. Rather than breaking the network up into array-sized pieces and evaluating each piece sequentially the system could instead use a small array-sized network to control access to an array-sized subsection of a larger network. Provided the subsections overlapped then the larger network should still be able to form a consistent ordered mapping despite the fact that it is only being accessed in a piecewise fashion.

5. Learning the Motion Map of a Robot Arm

The network developed by Saxon and Mukerjee [25] learns the motion map of a two-degree-of-freedom robot arm using a self-organizing feature map. This map can then be used for path-planning with obstacle avoidance.

5.1 System Operation

The system uses a 2-D self-organizing feature map of the type proposed by Kohonen (see Section 3). This map receives four inputs - two representing the cartesian-space position of the end-effector and two representing the joint-space position of the end-effector.

The system is trained by moving the arm to a random point in joint space, recording the cartesian position of the end-effector and then feeding both the cartesian and joint information to the network. After many thousands of such trials the system should learn a topological mapping of the input space. In essence each neuron comes to represent both a point in cartesian space and a point in joint space.

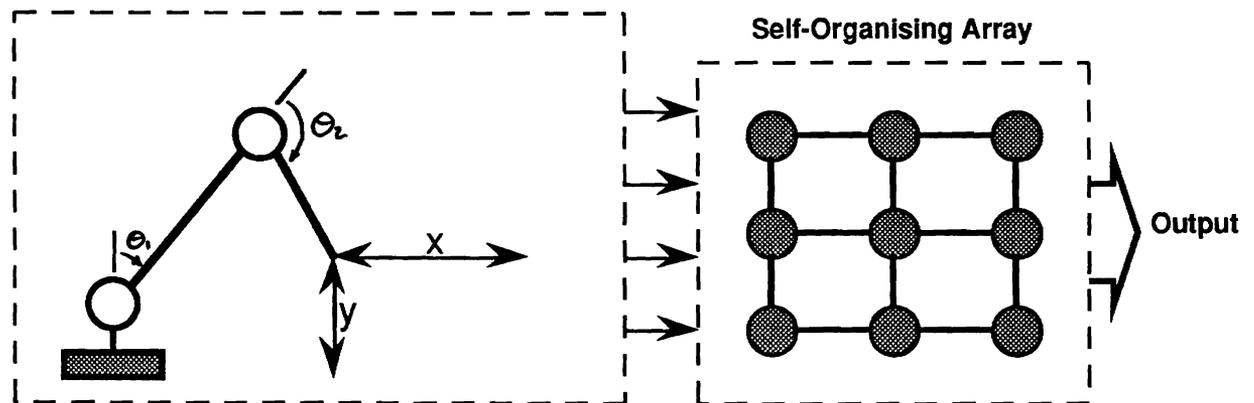


Figure 5.1 An overview of Saxon and Mukerjee's system.

5.2 Application to Path Planning

Once the motion map has been learned it provides the basis for a rather elegant means of planning trajectories around obstacles as described by Saxon and Mukerjee.

The basic idea is that each neuron in the 2-D net corresponds to both a joint-space and a cartesian space position. Performing trajectory planning using this net allows the incorporation of information from both joint and cartesian spaces while still only requiring that a 2-D array of possible paths be searched.

Obstacles can be incorporated by disabling the joint-space inputs to the network, applying the cartesian space position of the object(s) to the net and then disabling those neurons which produced maximal responses.

A possible trajectory which avoids these obstacles can then be determined by choosing a path through the 2-D neural array which starts at the neuron which most closely represents the current joint-space position of the arm, avoids any neurons which have been disabled and ends at the point which most closely represents the desired end-effector position (in either joint or cartesian space).

One way to accomplish this is to associate a score with each neuron in the network. If the maximum score were given to the destination neuron and if progressively lower scores were given to neurons further away in the array then a suitable path could be determined by selecting a path through those neurons with the highest scores. This is shown more clearly in Figure 5.2.

Saxon and Mukerjee suggest an alternative technique in which a path between the start and destination neurons could evolve through a process of "spreading activation". However, it is not clear how this could be achieved.

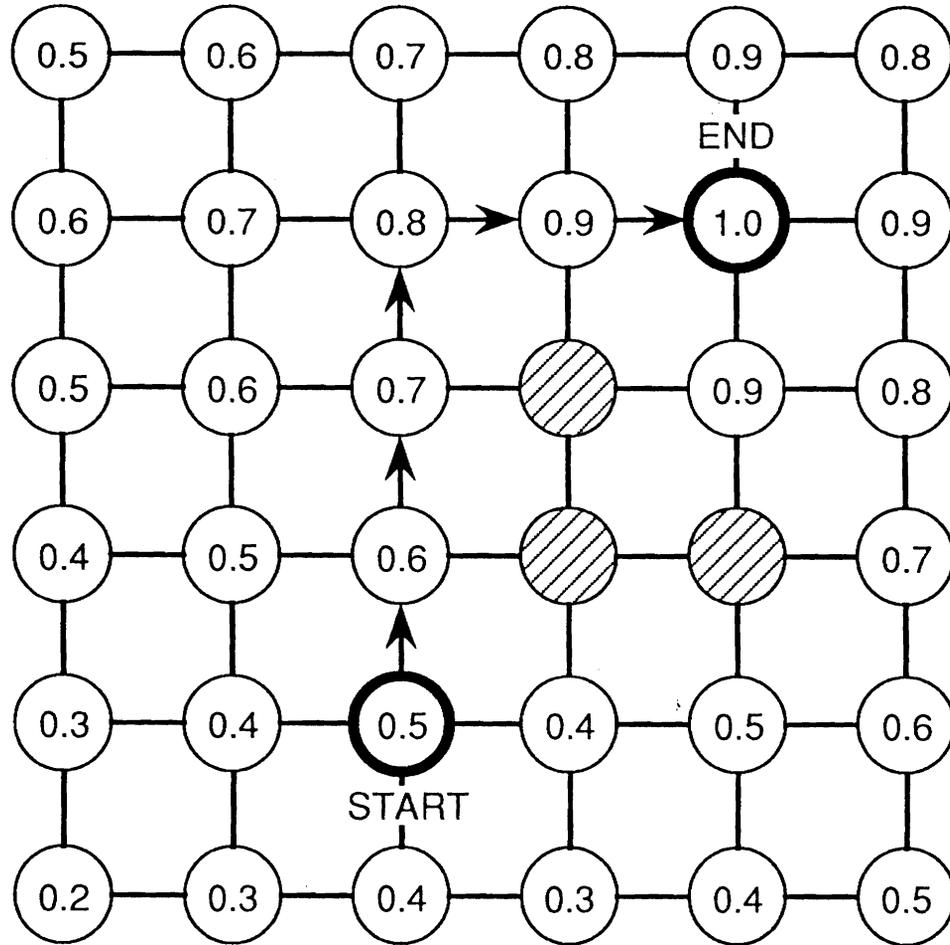


Figure 5.2 An example of how a trajectory planning scheme might operate by associating a score with each neuron. In this case the destination neuron was assigned a score of 1.0 and the activation levels of other neurons decrease linearly with distance through the array. Note that the system has correctly chosen a clockwise path around the obstacle (an anti-clockwise path would be longer). Note also that a square grid may not necessarily produce the best results and other connectivity options (for example arranging the neurons in a hexagonal structure) may produce smoother trajectories.

5.3 Discussion

5.3.1 Comparison with Kohonen's Work

This application appears very similar to the example presented by Kohonen (see Section 3.3). Indeed if one of the manipulators in that system were replaced by one which used prismatic joints which moved parallel to the cartesian axes then the systems would be identical.

5.3.2 Examination of Results

In order to examine some of the claims made in the paper the author attempted to duplicate their experiments. A two link manipulator (each link being one unit long) was simulated at randomly chosen positions in joint space. The cartesian position of the end effector, as well as the joint space configuration of the arm, were applied as input to a 2-D network. Initially the weights for each neuron in the network were set to small random values, the gain factor was set to 0.5 and the neighborhood size was a 7x7 square centered upon the maximally responsive neuron. As learning progressed the gain factor was reduced exponentially and the neighborhood size was decreased linearly. The results of this experiment are shown in Figure 5.3.

Examination of the final result shows that the system has correctly learned a mapping over both the joint and cartesian spaces. In particular the system appears to have coped well with the small area of the workspace where there exists some redundancy in the arm configuration.

It is interesting to note that these results appear much better than those obtained by Saxon and Mukerjee.

This is possibly due to different learning rates being utilized however it is difficult to test this hypothesis since they do not describe their learning algorithm in sufficient detail.

In their results they noted that "the entire configuration [joint] space can not actually be learned with a sheet of neurons" and claimed that this was caused by the fact that "the configuration space actually forms a torus". However, examination of Figure 5.3 shows that the system does appear to have learned the

entire joint space. Furthermore, since the joints are limited in their range of motion a planar representation would appear to be quite adequate.

They also suggest that the training strength of the edge neurons should be doubled "so that the edges of the network are pulled to the boundaries of the position map". This is not correct. In fact doubling the training strength of any subset of the network would have the undesirable effect of allowing an ordered network to become unordered.

An examination of the results of the path planning algorithm described by Saxon and Mukerjee is difficult because it is unclear from their paper whether the example they present is taken from an actual system or merely the result they might expect from a real system.

It could be predicted however that, while the described system will correctly locate a possible collision-free trajectory, it will not necessarily find a path which is optimal either in terms of travelling time or joint-space motion.

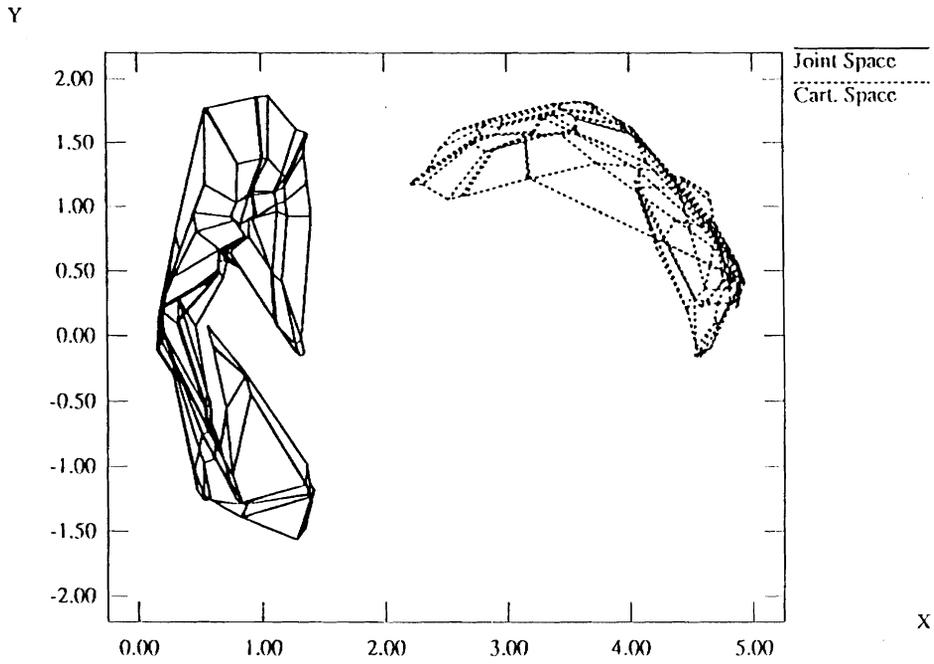
5.3.3 Alternative Implementations

In their system (and also in all of Kohonen's examples as well as in Martinez and Schulten's system) the initial neuron weights were given random values. However, if the user has some knowledge about the system then one could presumably make use of this in order to obtain a better result.

For example, if one knew the joint angle limits then the first two weight values could be chosen such that the network was evenly distributed across all possible joint values. These weights could then be frozen (held constant) while the system learned values for the remaining two weights based on the cartesian space positions. Once the learning approached a stable state the first two weights could be un-frozen to allow the system to adapt to future changes in both the cartesian and joint spaces.

An example of this type of learning is shown in Figure 5.4 where for the first 30,000 trials the system only modified the weights connected to cartesian space

a) Response after 200 trials



b) Response after 20000 trials

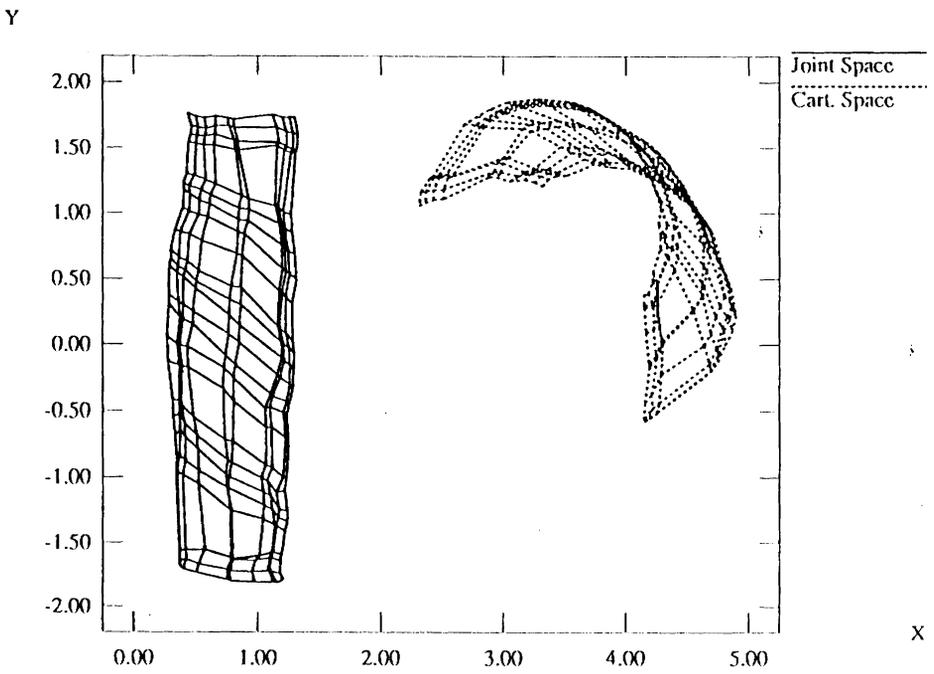
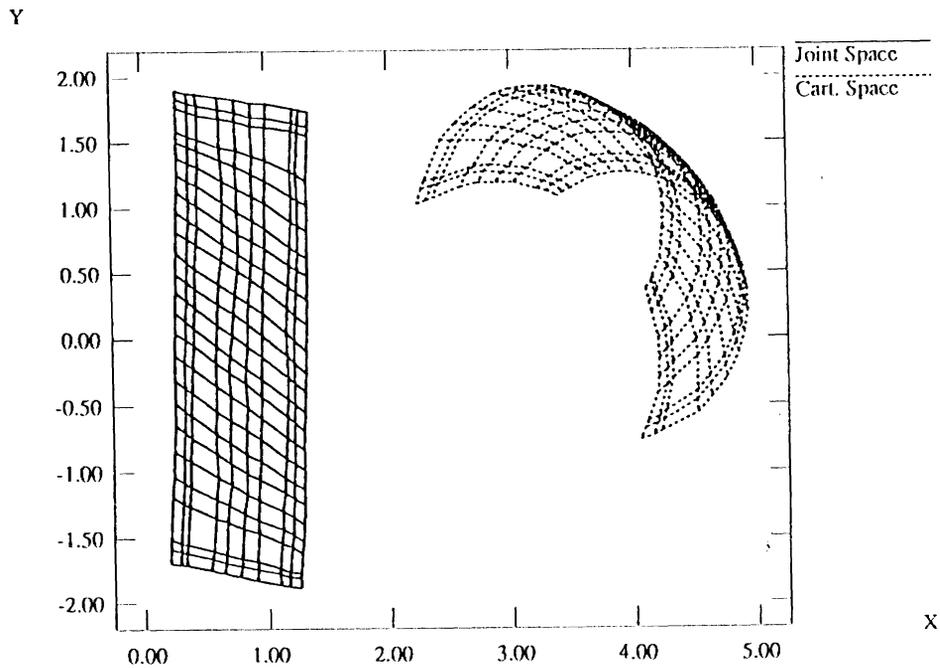


Figure 5.3 (continued on next page).

c) Response after 40000 trials



d) Response after 60000 trials

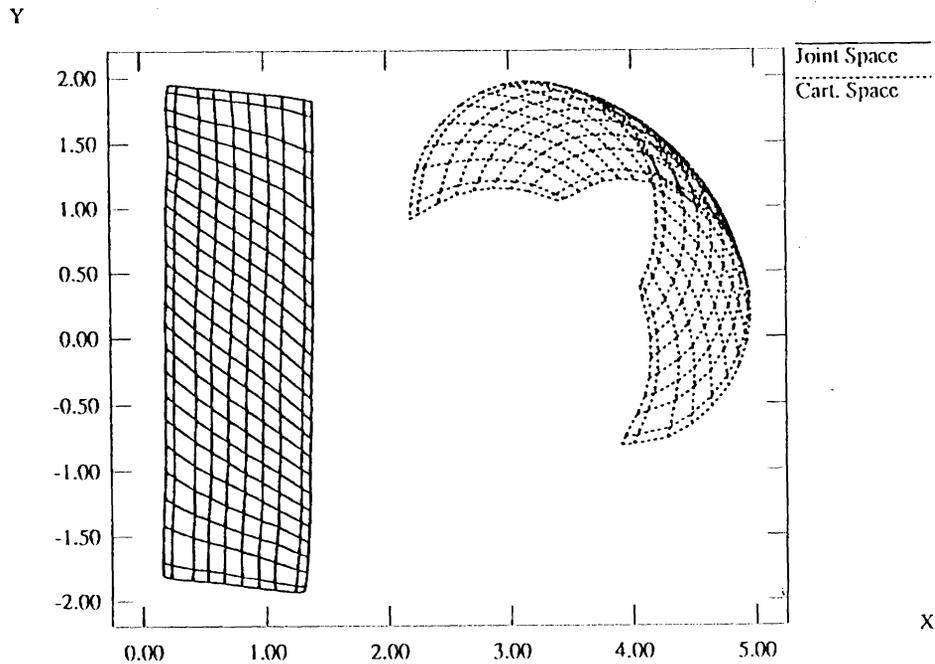
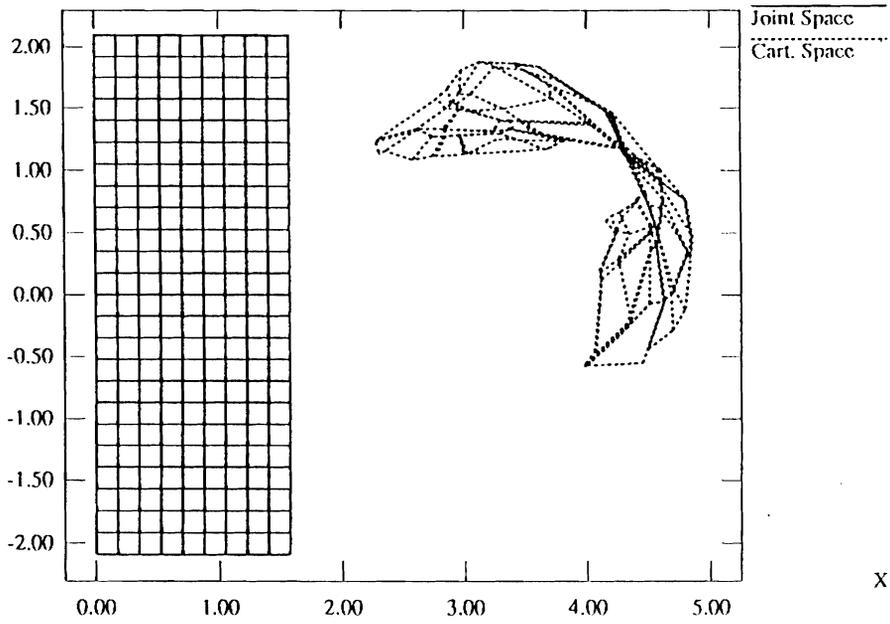


Figure 5.3 The positions in both joint and cartesian space to which each neuron learned to become maximally responsive. The origin of the cartesian space is at (3,0). The gain factor was reduced exponentially from 0.500 to 0.001 during learning while the neighborhood size was reduced linearly from 7x7 to 3x3.

a) Response after 200 trials

Y



c) Response after 20000 trials

Y

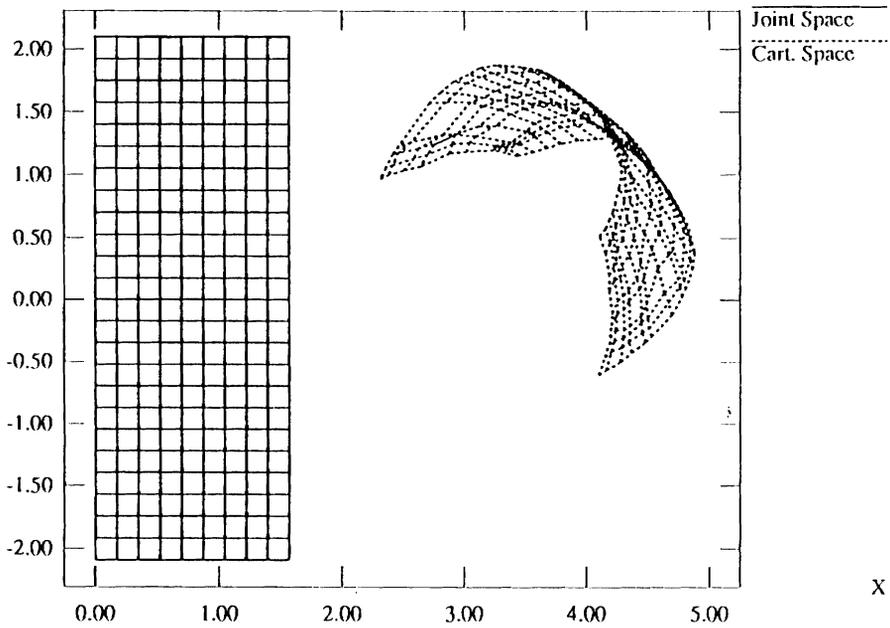
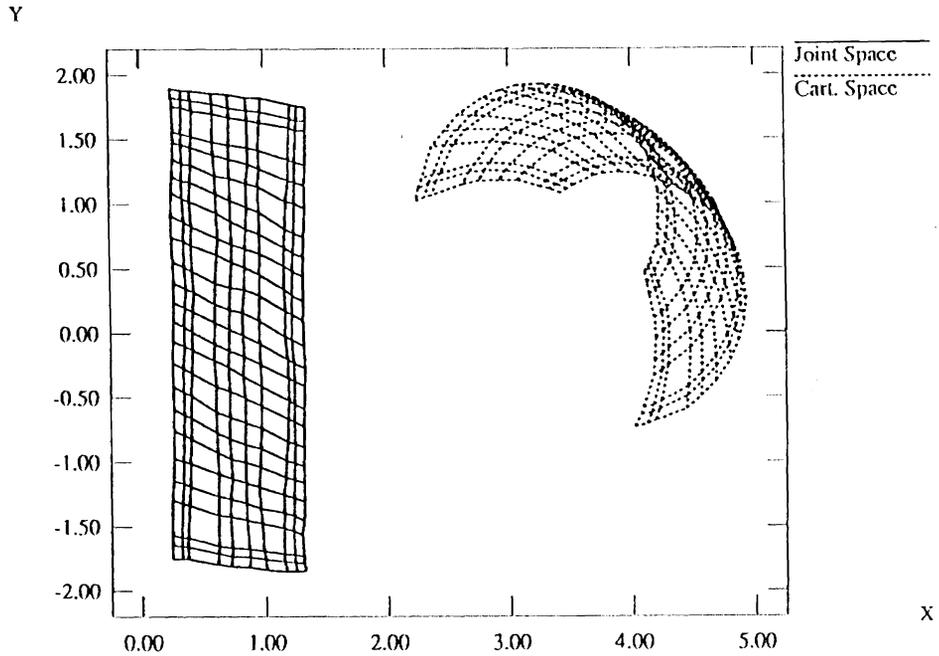


Figure 5.4 (continued on next page).

a) Response after 40000 trials



c) Response after 60000 trials

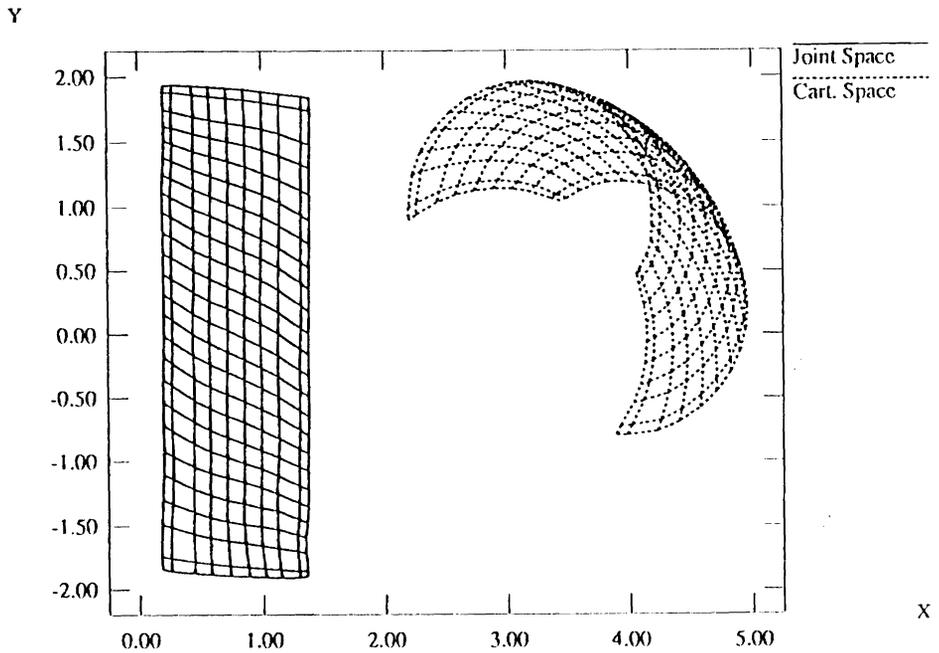


Figure 5.4 The positions in both joint and cartesian space to which each neuron learned to become maximally responsive. The origin of the cartesian space is at (3,0). The weights connected to cartesian inputs were frozen for the first 30,000 training cycles. The gain factor and neighborhood sizes were consistent with those used to produce Figure 5.3.

inputs. During the subsequent 30,000 trials the system was allowed to modify all four weights to achieve the result shown. This result isn't any more accurate than that obtained when all the weights had to be learned from scratch however it was achieved much more quickly since initially only two of the weights needed to be learned.

It might be tempting to leave the first two weights frozen however if this were allowed to happen then one of the significant advantages of this network - its ability to adapt - would be lost.

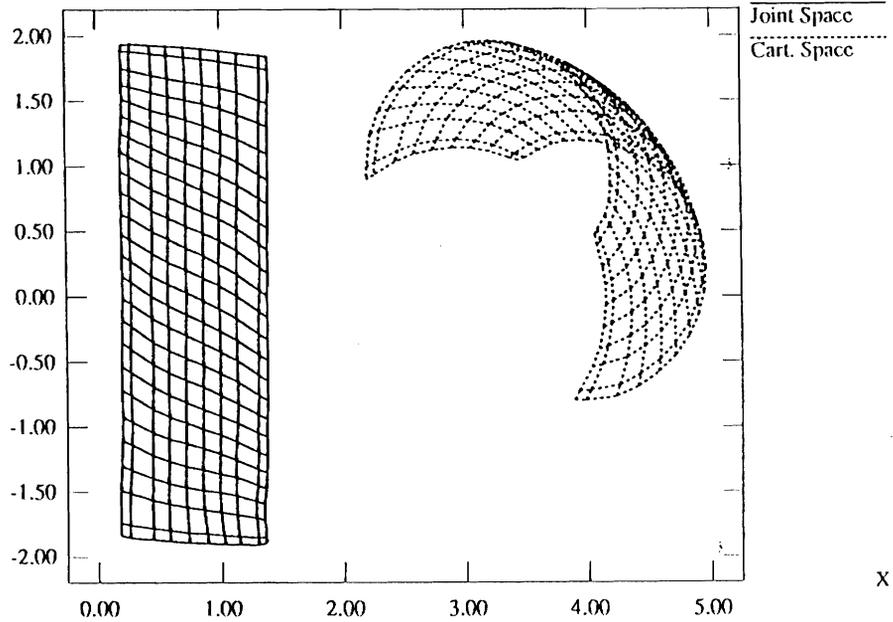
This ability to modify all input weights gives the system a certain plasticity and this allows it to conform to changes in the environment. For example if the robot were prevented from moving in the region where $y < 0$ then it can adapt as shown in Figure 5.5.

The system can also adapt to account for cases where the joint-space positions are not evenly distributed. For example, consider the case where on every tenth trial the joint space point, rather than being chosen to lie randomly anywhere within the allowable space, was instead chosen to lie on the line $\theta_1 = \theta_2$ in the interval $0 < \theta_1 < \Pi/3$. The results of this are shown in Figure 5.6.

Examination of the results shows that the system has distorted the array to increase the number of neurons which correspond to points near that line. This would have the practical effect of increasing the relative accuracy of the systems "knowledge" near that area. In effect the system is able to increase its accuracy near commonly travelled trajectories at the expense of a decreased accuracy in other areas.

a) Initial Response

Y



b) Response after 400000 trials with restricted motion

Y

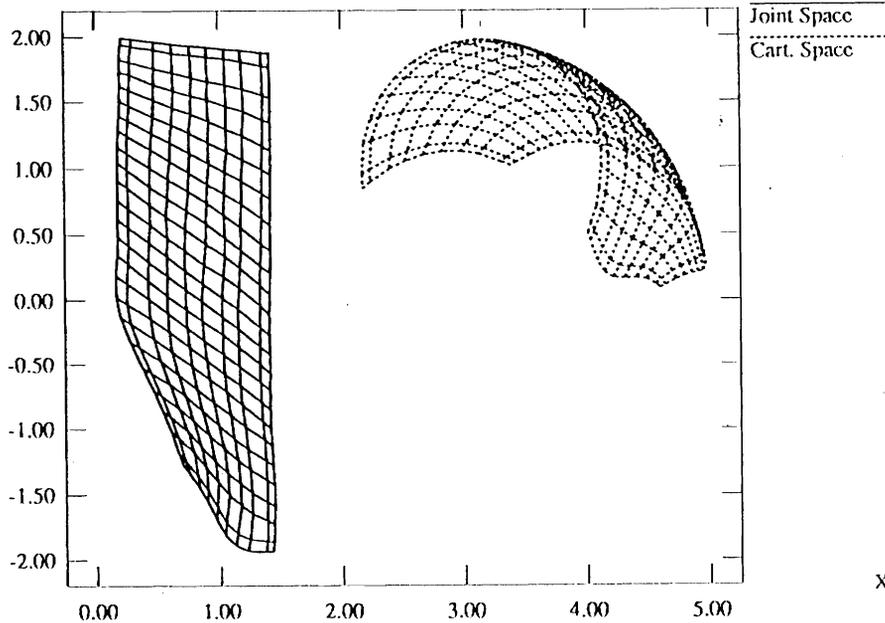
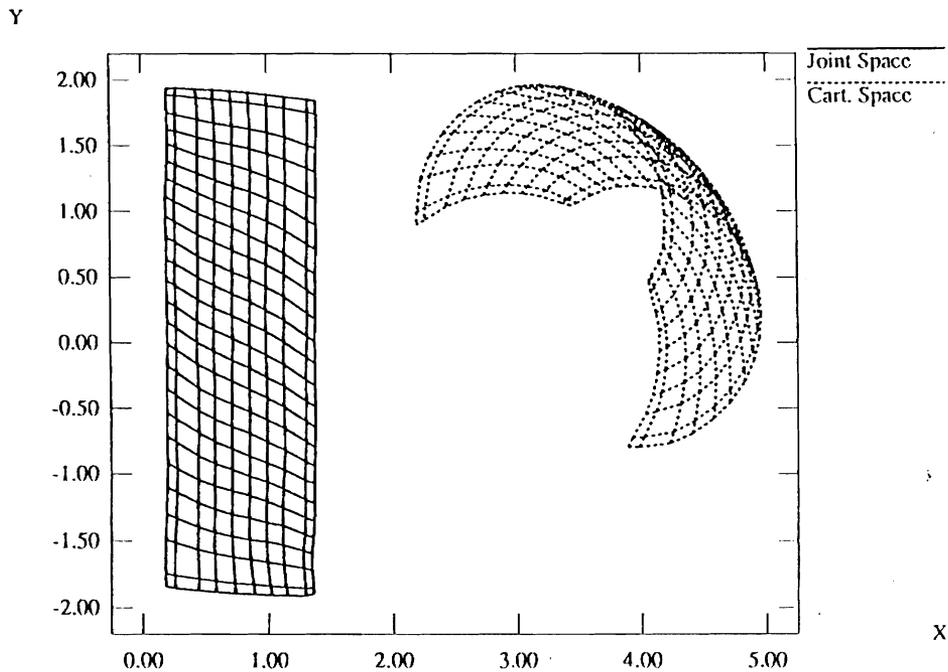


Figure 5.5 The positions in both joint and cartesian space to which each neuron learned to become maximally responsive. The system was initially trained as described in Figure 5.3 then the gain factor was fixed at 0.005, the neighborhood size was set at 3x3 and cartesian motion was constrained to the region $y > 0$.

a) Initial Response



b) Response after 20000 trials with motion along a straight line in joint space

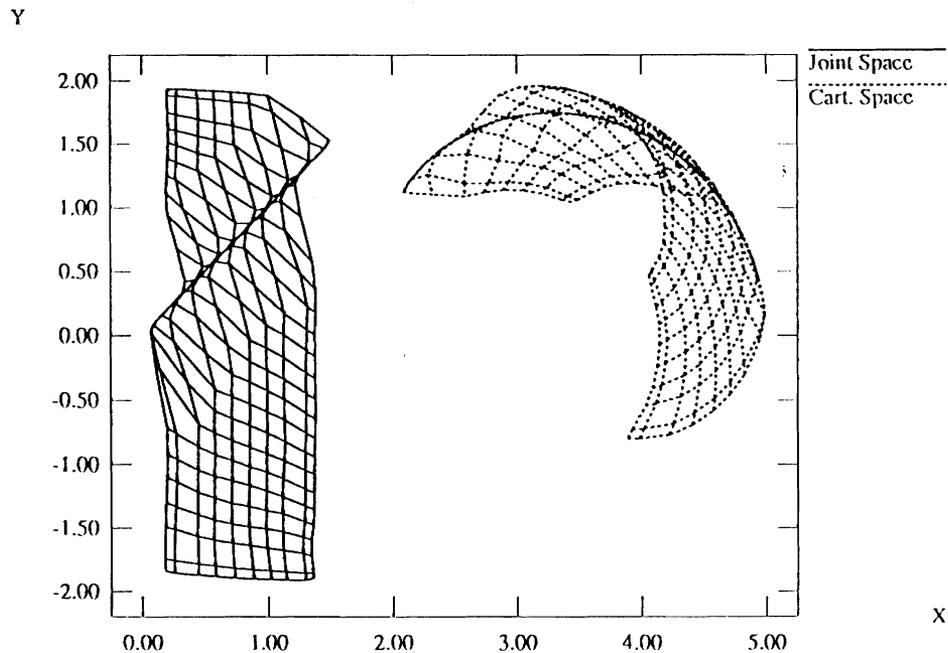


Figure 5.6 The positions in both joint and cartesian space to which each neuron learned to become maximally responsive. The system was initially trained as described in Figure 5.2 then the gain factor was fixed at 0.002, the neighborhood size was set at 3x3 and joint positions were constrained to lie along the line $\theta_1 = \theta_2$.

6. Conclusions

The adaptability inherent in Kohonen's self-organizing feature maps provides the ability to learn abstract representations of systems in which the relationships between the inputs are not known. Also the plasticity inherent in this type of network allows it to adapt to changes in the environment.

However these advantages come at a cost. Self-organizing feature maps will generally require more computing power than conventional methods and the lack of a strong mathematical foundation means that there is no guarantee that real world systems will always maintain a consistent ordered mapping. The absence of a rigorous mathematical treatment also means that parameters must be selected in a somewhat ad hoc manor and this may prove difficult in cases where the desired result is not well defined.

Kohonen presented two examples which show the ability of self-organizing feature maps. In the first a single feature map was used to recognize phonemes in connected speech. The ability of this system to compress a fifteen-dimensional input into a two-dimensional representation is particularly impressive. The second example used a network to learn the relationship between the joint-space parameters for two connected robot arms.

Martinez and Schulten have described a system which makes use of a hierarchical structure composed of many self-organizing feature maps. While it was noted that there may be some practical problems, the general idea of using a hierarchical structure appears sound and may be applicable to a wider range of problems.

The robotics application using a single self-organizing feature map which Saxon and Mukerjee describe appears similar to the robotics application described by Kohonen. The results presented in their paper appear inferior to those which the author obtained while attempting to duplicate their work however the lack of specific information about their learning algorithm makes it difficult to trace the source of these discrepancies. The use of the system they describe should simplify path planning by combining multiple constraints into a 2-D structure.

7. References

- 1 Kohonen, T., "Representation of Sensory Information in Self-Organizing Feature Maps, and the Relation of these Maps to Distributed Memory Networks, Computer Simulation in Brain Science, R.M.J. Cotterill (editor), Cambridge University Press, Great Britain, 1988, pp12-25.
- 2 Kohonen, T., "Self-Organizing Feature Maps and Abstractions", Artificial Intelligence and Control Systems of Robots, I. Plander (editor), Elsevier Science Publishers, The Netherlands, 1984, pp 39-45.
- 3 Martinez, T.M. and Schulten, K.L., "Hierarchical Neural Net for Learning Control of a Robot's Arm and Gripper", IEEE INNS Joint Conf. Neural Networks, Vol. II, 1990, pp 745-752.
- 4 Saxon, J.B. and Mukerjee, A., "Learning the Motion Map of a Robot Arm with Neural Networks", IEEE INNS Joint Conf. Neural Networks, Vol II, 1990, pp 777-782.
- 5 Hofstadter, D.R., Godel, Escher, Bach: An Eternal Golden Braid, Penguin Books, Great Britain, 1979, pp 337-365.
- 6 Kohonen, T., "Representation of Sensory Information in Self-Organizing Feature Maps, and the Relation of these Maps to Distributed Memory Networks, Computer Simulation in Brain Science, R.M.J. Cotterill (editor), Cambridge University Press, Great Britain, 1988, p14.
- 7 Kohonen, T., Self-Organization and Associative Memory (Second Edition), Springer-Verlag, Germany, 1987, p221-222.
- 8 Kohonen, T., "Self-Organizing Feature Maps and Abstractions", Artificial Intelligence and Control Systems of Robots, I. Plander (editor), Elsevier Science Publishers, The Netherlands, 1984, p 41.
- 9 Kohonen, T., "Self-Organizing Feature Maps and Abstractions", Artificial Intelligence and Control Systems of Robots, I. Plander (editor), Elsevier Science Publishers, The Netherlands, 1984, p 42.

- 10 Obermayer, K., Ritter, H. and Schulten, K., "Large-Scale Simulation of Self-Organizing Neural Networks: Formation of a Somatotopic Map", *Parallel Processing in Neural Systems and Computers*, R. Eckmiller (Editor), Elsevier Science Publishers, The Netherlands, 1990, pp71-74.
- 11 Kohonen, T., "Representation of Sensory Information in Self-Organizing Feature Maps, and the Relation of these Maps to Distributed Memory Networks, *Computer Simulation in Brain Science*, R.M.J. Cotterill (editor), Cambridge University Press, Great Britain, 1988, pp12-25.
- 12 Kohonen, T., "Self-Organizing Feature Maps and Abstractions", *Artificial Intelligence and Control Systems of Robots*, I. Plander (editor), Elsevier Science Publishers, The Netherlands, 1984, pp 39-45.
- 13 Kohonen, T., *Self-Organization and Associative Memory (Second Edition)*, Springer-Verlag, Germany, 1987, p60.
- 14 Kohonen, T. "Speech Recognition based on Topology Conserving Feature Maps", *Neural Computer Architectures: The Design of Brain-Like Machines*, I. Aleksander (editor), MIT Press, USA, 1989, pp 26-40.
- 15 Kohonen, T., "The Neural Phonetic Typewriter", *IEEE Computer Magazine*, March, 1988, pp 11-22.
- 16 Kohonen, T., *Self-Organization and Associative Memory (Second Edition)*, Springer-Verlag, Germany, 1987, pp143-157.
- 17 Kohonen, T. "Speech Recognition based on Topology Conserving Feature Maps", *Neural Computer Architectures: The Design of Brain-Like Machines*, I. Aleksander (editor), MIT Press, USA, 1989, pp31-32.
- 18 Kohonen, T., "Representation of Sensory Information in Self-Organizing Feature Maps, and the Relation of these Maps to Distributed Memory Networks, *Computer Simulation in Brain Science*, R.M.J. Cotterill (editor), Cambridge University Press, Great Britain, 1988, pp13-21.
- 19 Kohonen, T., "The Neural Phonetic Typewriter", *IEEE Computer Magazine*, March, 1988, pp 15-17.

- 20 Kohonen, T., *Self-Organization and Associative Memory (Second Edition)*, Springer-Verlag, Germany, 1987, pp122-127.
- 21 Acker, R and Kurtz, A, "On the Biologically Motivated Derivation of Kohonen's Self-Organizing Feature Maps", *Parallel Processing in Neural Systems and Computers*, R. Eckmiller (Editor), Elsevier Science Publishers, The Netherlands, 1990, pp229-232.
- 22 Martinez, T.M. and Schulten, K.L., "Hierarchical Neural Net for Learning Control of a Robot's Arm and Gripper", *IEEE INNS Joint Conf. Neural Networks*, Vol. II, 1990, pp 745-752.
- 23 Martinez, T.M., Helge, J.R. and Schulten, K.J., "Three-Dimensional Neural Net for Learning Visuomotor Control of a Robot Arm", *IEEE Trans. Neural Networks*, Vol.1, No.1, March 1990, p 133.
- 24 Martinez, T., Ritter, H. and Schulten, K., "Learning of Visuomotor-Coordination of a Robot Arm with Redundant Degrees of Freedom", *Parallel Processing in Neural Systems and Computers*, R. Eckmiller (Editor), Elsevier Science Publishers, The Netherlands, 1990, pp 431-434.
- 25 Saxon, J.B. and Mukerjee, A., "Learning the Motion Map of a Robot Arm with Neural Networks", *IEEE INNS Joint Conf. Neural Networks*, Vol II, 1990, pp 777-782.