A Digital Learning System For Tracking Pattern Features

By

A. P. Reeves

A Thesis Submitted for the Degree
of Doctor of Philosophy at the
University of Kent at Canterbury

1973

CONTENTS

ACKNOWLEDGEMENTS

I wish to acknowledge the support that the Science Research Council has given to this project.

I would like to thank Dr. I. Aleksander for his help and supervision, and also the staff of the Electronics Laboratories at the University of Kent for their advice and for providing the research facilities.

I am especially grateful to my colleagues and friends in the Electronics Computer Laboratory for their help which has enabled me to complete this thesis.

Finally, I would like to thank Miss D. Paine for the care she has taken in typing this thesis.

A. P. REEVES.

ABSTRACT

A novel pattern processing scheme has been investigated which makes use of the motions generated by a window which tracks the lines or contours of a digitised television image of a black/white pattern. The novel features of the proposed scheme are that adaptive learning networks are used for both tracking and classifying. The tracking strategy is learnt from a human teacher.

Here one combines two methods of machine pattern recognition which, in isolation, have a limited performance. These are,'static learning networks'which have known limitations, and 'programmed tracking systems' in which the pre-programming itself may be limiting. In this combination one avoids some limitations of these systems because pre-programming of strategies is not necessary and feedback exists to make the task of the nets a dynamic one.

The thesis describes a hardware visual input and a special-purpose software system which were developed for this investigation. Also, several new modifications of the SLAM (Stored Logic Adaptive Microcircuit) element are discussed.

Beyond its practical application it is possible to conclude that the system developed here may be useful in the study of hypotheses regarding living animal systems which involve eye movements.

CHAPTER 1

INTRODUCTION

1.1 STATEMENT OF THE PROBLEM.

The problem investigated in this thesis represents one aspect of pattern recognition by a machine, and involves the learning of tracking strategies by means of adaptive digital networks.

The best general system in existence for recognising complex patterns is still to be found in man and animals. Consider what happens when we look at a complex pattern such as a photograph or a line diagram. The eye, which can only perceive accurately a small amount of detail at a time, roves over the pattern and enables the brain to gather information from the pattern sequentially.

Now consider the ways that machines have been used to recognise patterns. Some of these compare the whole of the unknown pattern with a set of templates, but this method is usually discarded in favour of more powerful machines which can examine parts of the input pattern in detail. Often, in these machines, this examination results in a sequential description of the pattern.

Another approach to machine pattern recognition has been made with 'adaptive learning networks' these learning networks are often considered as a step away from normal computing processes towards the processing of information in a way similar to that of brains. One method used in practice involves the sampling of the whole of the input pattern with the elements of the learning network. One arrives at a final classification on the basis of a combinational decision made on the outputs of these elements. The theoretical limitations of this method(33), as with the limitations of the template matching method, are now well understood.

It is proposed in this thesis to investigate the effect of allowing the input matrix of a learning network to rove over a pattern so that parts of the pattern may be examined in detail, as is the case with the eye. One notes that such a system would have to be able to direct its own roving activity as well as analyse the data it receives. Two important topics require attention in the context of this system.

The first deals with adaptation to the attention shifts that are necessary for recognising a pattern and the organisation of a learning network aimed at achieving this task.

Second, one asks how a learning network can be organised to analyse the sequential data which it receives from the 'roving eye'.

This thesis is primarily concerned with these two questions.

In considering a pattern recognition system of this form, the following three main areas of research are relevant.

Character recognition: This is a restricted case of pattern recognition where only characters are considered.

Scene analysis: This is a more general case of pattern recognition and is usually concerned with three dimensional scenes or two dimensional images of these scenes.

Visual Form perception: Where the perception of the shapes of objects by man and animals is considered.

Pattern recognition is generally tackled by making a classification based on a set of selected measurements extracted from an input pattern. These selected measurements called 'features' are supposed to be less sensitive with respect to the variations encountered within classes, and also to contain less redundancy.

A general scheme for a pattern recogniser is shown in Fig. 1.1.1. The input pattern is transformed by a preprocessor which operates in a uniform way over the input pattern to present the data in a form more acceptable to the feature extractor. The feature extractor takes the selected measurements mentioned above. The classifier decides to which class the input pattern belongs on the basis of the values of the extracted features. This model is not a necessary construction for a pattern recogniser but is used,

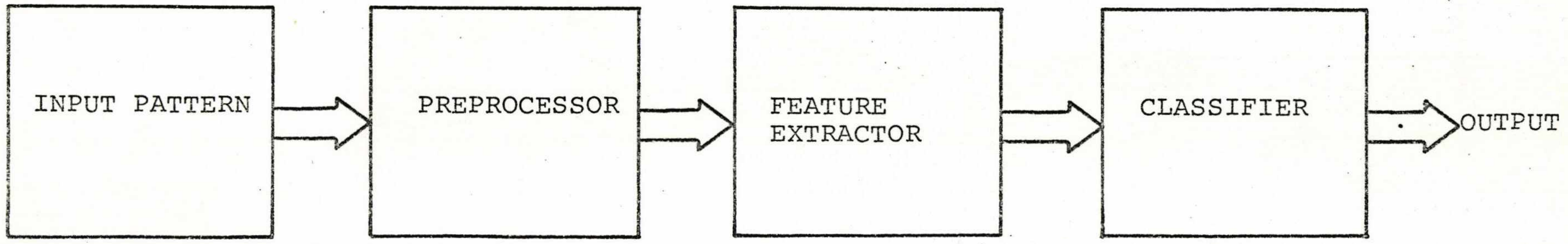

INPUT PATTERN
PREPROCESSOR
FEATURE EXTRACTOR
CLASSIFIER
OUTPUT

FIGURE 1.1.1

for convenience, to standardise a framework on which to base one's discussions.

1.2 CHARACTER RECOGNITION.

Character recognition is the particular case in pattern recognition when recognising characters only is considered. Much work has been done in this field due to the need for machines to be able to read normal written text. Most of this work is of a specialised nature due to the limited, well defined class of inputs which are to be detected. The following account does not mention all the methods used in character recognition but only those which lead towards this project.

Character recognition is characterised by having a limited set of characters to be recognised. The input pattern is usually in the form of a two dimensional binary matrix representing the character.

1.2.1 Template Matching Pattern Recognisers.

Possibly the simplest, and most obvious, form of character recognition which has been used is template matching. With this method no preprocessing or feature extraction occurs and the raw data is input directly to the classifier. The classifier compares the input with a template of each class, and the classification is based on a preselected matching criterion or similarity criterion. Such a simple method has many limitations. It is sometimes difficult to select a good template from each class and to define a proper matching criterion. Also, this method is both size limited and font limited.

Many variations of this method have been investigated. For example, the method described by Taylor 1968 (50) in which a mask with analog weights is used. However, all these methods are restricted by the limitations mentioned above. Owing to the obvious drawbacks, template methods were not considered in this project.

1.2.2 N-Tuple Pattern Recogniser.

The n-tuple recognising method may be considered as the next development in complexity. This method considers sets of n-tuple samples from the input space. These n-tuples are usually selected randomly or by some adaptive algorithms. Early work on this method was done by Bledsoe and Browning 1959 (5) in which a binary matrix is used for each class to store all the possible n-tuple states. This is initially set to zero and during training 1's are put into the elements of this matrix which are selected by the input pattern. When an unknown pattern is presented to the system the number of 1's which this pattern references in each matrix is noted. The matrix in which the highest number of 1's are referenced decides the class of the input. This system was further developed by Bledsoe and Bisson 1962 (6) in which analog matrices are used and the frequency of occurrence of n-tuples during training was also considered. Work with similar structures has since been investigated by

Ullman 1969 (54) where the optimum size of the n-tuple was found for a limited number in the training set and Chung 1973 (7) in which the optimum amount of teaching and other features are considered. Character recognition by the n-tuple method often suffers from the same limitations as a perceptron,(Rosenblatt (41)). A full, detailed, analysis of the theoretical topological properties and limitations of perceptron structures is discussed by Minsky and Papert 1969 (33). Learning networks consisting of binary n-tuple sampling elements (SLAMs) have been used in this project. However, feedback has been introduced around the learning networks to overcome some of the limitations referred to above.

1.2.3 Scanning Pattern Recogniser.

The description of a pattern may be generated by a list of localised features of the pattern in which the position of these features is also indicated.

A practical system which embodies this method is the 'Scan Data Optical Reading System' (46) which is capable of reading multi-font characters and a limited set of hand-printed characters. The input pattern is presented on a 40x30 bit binary matrix obtained from a flying spot scanner and algorithms are used to normalise, centralise and adjust the threshold for the sampling brightness level to compensate for variations in the background. The feature extractor then scans the input pattern with a feature window and compares each input of the feature window with a list of 400 possible features using a mask matching criterion. The input pattern is divided into 9 areas and the area in which each feature is found is also recorded. Hence after scanning, the feature extractor outputs a set of detected features with their approximate positions on the input pattern. The classifier compares this feature/position list with a set of feature/position lists to classify the input character. The scanning method is inherently slower than the parallel methods mentioned previously. However, high speed is not essential to this system as the data inputting hardware has a maximum speed of 800 characters per second.

Hunt 1972(25) describes a system in which the input pattern is scanned with a set of logical operators. These logical operators consist of a ternary mask which contains 1's, O's and spaces. A match is made if the 1's and O's coincide with a part of the input patterns exactly. A 5x5 result matrix (which corresponds to 25 equal areas of the input pattern) is generated in which matches of the operators are recorded. Only one match can be recorded in each cell of the result matrix and a priority scheme decides the value of the cell if several different matches occur in its area. The result matrix is used to describe the input pattern and is input to the classifier.

A different approach, but still using a scanning method is described by Hosking and Thomson 1968(22) and is further developed by Hosking 1972(23). In their method, the input pattern is scanned once and the feature extractor indicates when features occur. Features are of a general type e.g. the start of a line, the join between two lines etc. and about 10 different types of feature are detected. The detected features are numbered from 1 to n as they occur and a connection list stating which features are joined, is generated. Hence, the description of the pattern which is presented to the classifier is an ordered list of connected features (the type of the features is not specified in this list).

Uhr and Vossler 1961(53) have also described a system in which the input pattern is scanned by a set of logical operators. The feature extractor forms a description of the pattern by noting the number of matches for each operator and also the average of the coordinates of these matches. For this system, Uhr and Vossler have developed algorithms for generating operators and for evaluating their performance. This enables the system to adapt to different styles of inputs and to improve its own performance.

Although the above systems have sequentiality and window extraction in common with the method in this thesis, the scanning is not determined by the "seen" elements in the window. Such systems are considered next.

1.2.4 Tracking Pattern Recognisers.

The tracking method uses the fact that characters are formed by a set of lines. If all the lines of an input pattern are tracked then all the localised features of the pattern will be encountered, hence, it is not necessary to scan over all of the input pattern.

Most tracking systems have the following general scheme. The input pattern is thinned so the lines are only one or two bits wide; then a tracking algorithm is used to track along these lines. The tracking motions produced during the tracking are then input serially to the feature detector which uses an algorithm to change these motions into an ordered list of features. Features are usually defined to be line segments, curves and line junctions etc. Either in the preprocessor or in the feature detector, a smoothing algorithm is used so that small variations in the shape of the input pattern will not be interpreted as features.

It is also possible to track the edges of the input pattern rather than the lines. This has the advantage that the character does not have to be thinned and that a simple operator may be used to track the edge. An example of such a system is described by Saraga et al 1967(43) and a further development of the tracking operators is described by Saraga and Wavish 1971(44). A different approach to edge following is described by Taussaint and Donaldson 1970(52). In their method the input pattern is divided into several areas and the maximum and minimum positions reached while tracking in each area is recorded to describe the input pattern. Mason and Clemens 1968(32) used a similar approach which involved detecting maxima and minima when the edge of the pattern is tracked. Their system represents a character by a binary codeword which is generated by forming a string adding a 1 whenever an X coordinate maximum or minimum occurs, and a 0 whenever a Y coordinate maximum or minimum occurs.

As far as preprocessing is concerned, a scanning method of thinning lines is described by Saraga and Woollons 1968(45) and a review of these techniques is given by Deutsch 1968(9). Some interesting preprocessing schemes may be achieved if the input matrix is hexagonal rather than square and this is discussed by Golay 1969(18) though this is more useful in cases other than character recognition where the input pattern is not uniquely orientated.

Grimsdale et al 1959(19) have designed a system which splits the input patterns into 'regions' which includes line and curve segments and the shape of junctions. A method of extracting features such as line endings, change of direction, junction of lines, etc. is described by Parks 1969(36). This system has been further developed by Watt and Beurle 1971(55) in which these are ordered and then the ordered feature list is classified. Deutsch(9) also describes a system which forms an ordered feature list relating to a skeleton shape preprocessed from the original input pattern.

Finally, Eden 1968(12) describes a method for recognising cursive handwritten script by splitting the script into a sequence of 'strokes'. He defines a set of 28 different types of strokes though, in fact, only nine of these are used for English script.

The system described in this thesis is designed to recognise patterns by tracking them. However, it differs from the general form for tracking systems in two main ways. Firstly, the tracking strategy is taught to the system by a human teacher and secondly, no preprocessing (such as line thinning, etc.) is applied to the input information before it is presented to the tracking system.

1.3 SCENE ANALYSIS.

In scene analysis three dimensional scenes are usually considered though they are reduced to two dimensional images. The image is formed by a two dimensional array of picture elements which have a brightness level, obtained from the scene, associated with them. The task in scene analysis is to detect and recognise from this image the objects present in the scene and the positional relationships between them. A good review of the techniques which have been used to achieve this is given by Duda and Hart 1973 (11).

For the purpose of this project, the methods of interest are those which operate on the image to reduce it to a line drawing (this is called spatial differentiation) and then reconstruct those lines to determine the original objects.

An example of such a system is described by Forsen 1968 (15) where the image is first spatially differentiated. Then the resulting binary matrix is scanned by a set of 7x7 bit feature matrices and a new matrix is generated indicating where and what type of features have been detected. A line following algorithm could then be used to track this matrix.

An interesting system is described by Symons 1968(49) in which contour detecting and following are simultaneously conducted on the image which has not been spatially differentiated. Also, features may be detected as the tracking proceeds.

A similar method is used by Pingle 1969(37) who describes a system which is designed to rapidly trace the outline of an object. (This system is intended for real-time manipulation of mechanical arms and hence, must be as fast as possible.) After tracking an algorithm is used which assumes that the object is made of straight lines and attempts to locate the positions of the corners.

A system described by Ledley 1964(29) uses a 'bug algorithm' to track the boundaries of digital images of chromosomes in a manner similar to that used in character recognition. From the tracking motions the chromosome is described by a sequence of boundary segements. Kelley 1971(27) has developed a system which uses 'planning' to extract the contour of a head from a photograph. Planning involves reducing the size of the image and firstly tracking the contours of this. Then this rough tracking is used as a plan for tracking contours of the original image.

The system described in this thesis could be used for scene analysis of a spatially differentiated scene. It differs from previously mentioned methods in that it uses a learnt adaptive tracking strategy.

1.4 THEORIES AND MODELS OF VISUAL PERCEPTION.

Much work has been done on trying to determine the mechanisms of visual perception in man and in animals. Although much progress has been made, especially with regards to determining the functions of the retina of the eyes and the optic nerve, still very little is known about the method of 'classification' which exists in the brain to enable it to perceive objects. For further details in this field Kolers 1968(28) reviews some of the physiological aspects of pattern recognition, and a review of the neurophysiology of the visual system is given by Chung 1968(8). A detailed review which covers the whole field of the perception of form is given by Zusne 1970(57).

Of particular interest is the research into the movements of the eye with respect to recognition. A review of the experiments conducted with measurements of eye movements, including a description of the techniques involved in obtaining these measurements, is given in Zusne(57). One set of experiments, conducted by Yarbus 1967(56), is to measure the eye movements when subjects view a two dimensional art work. His results show that, when looking at pictures, the observers fixate more frequently the features which are actual or potential carriers of information. Noton and Stark 1971(35) have developed a theory of perception (see section 1.4.2) in which the order of the fixations is important and their experimental results show that, when viewing a simple line drawing picture, repeated sequencies of fixations and attention shifts do tend to occur.

1.4.1 The Eye and the Visual Pathway.

A simplified description of the retina and visual pathway of the eye is as follows. The light coming through the lens of the eye falls on a mosaic of receptor cells (rods and cones) in the retina. The receptor cells connect with bipolar cells and the bipolar cells connect with retinal ganglion cells which send their fibres (i.e. the optic nerve fibres) to the lateral geniculate body which transfers this information to the visual area of the cerebral cortex (the visual cortex).

There are two kinds of light receptors; rods, which are very sensitive to blue-green light and cannot be used for colour, and cones, which are 1000 times less sensitive to light than rods, they are sensitive to colour but are useless in poor lighting conditions. There are about 120.10^6 receptors in the retina and about 10^6 fibres in the optic nerve, a reduction of about 120:1. The cones are represented most densely in the centre of the eye and the rods most densely in the periphery. In general, cones tend to have direct lines (via bipolar cells) to ganglion cells whereas many rods converge upon a single ganglion cell. The fovea (the centre of the eye) only contains cones, and there is a 1 to 1 relation between receptors and ganglia in this region.

Hence, when the eye observes a scene under normal lighting conditions, the optic nerve transfers a great amount of detailed information about a small area at the centre of the field of view together with a general impression from the rest of the field of view.

The ganglion cells perform the first stage of information processing. The input of a ganglion is taken from a localised area of the retina called its receptive field. The output from the ganglion depends on the pattern of light in its receptive field. Much work has been done on determining the functions which these ganglia perform. Many experiments have been conducted with animals and these have shown that there is no general law which holds across species with regards to the visual system. Hence, results obtained with animals are directly relevant to that species of animal only.

An interesting example is the frog for which four different types of ganglion cells have been identified (Lettvin et al 1959[31]). Ganglia of each type are grouped together and they map the retina continuously onto a single sheet of endings in the tectum of the frog's brain. Herscher and Kelly 1963[21] and Sutro 1968[48] have made hardware models of the frog's retina which contain replicas of the four different types of ganglia and are organised in a similar way to that found in the frog. One of these ganglion cells (usually called the bug detector) is sensitive to small dark convex objects which move centripetally with respect to its receptive field. Arbib 1971[2] presents the theory that the tectum of the frog behaves as a somatotopically organised parallel decision mechanism which enables the frog to perceive a fly and snap at it with its tongue.

Another example is the pidgeon which has ganglion cells that are sensitive to movements in one direction and not the other. A hardware model of this retina has been made by Runge et al 1968[42].

The functions of the ganglion cells depend on the particular animal and are often directed towards the particular needs of that animal. For mammals only two general types of ganglia cells have been detected. Most of this work has been done with cats but it is reasonable to assume that the same may in this case be true for man. These cells have circular receptive fields, one type contains a centre region which is excitatory and an outer region which is inhibitory and the other type contains the opposite.

Hubel and Wiesel 1966[24] have found that cells in the visual cortex of the cat have receptive fields which correspond to localised regions of the retina. These cells can be divided into two types: simple and complex. The simple cells are sensitive to spots of light (i.e. they have a similar function to the ganglion cells) or are sensitive to an edge of light in which the orientation of the edge is important. The complex cells are sensitive to both the form of an object and the position of the object with respect to the receptive field of the cell. Hubel and Wiesel used a rectangle as an object for these tests in which the dimensions, orientation and position could be varied.

A model of the processing part of the brain has been proposed and simulated by Kabrisky 1966(26). His contention is that the cortex behaves as a generalised planar (two dimensional) pattern manipulator which has a memory. He has designed an element the 'Basic Cortical Computational Element' which in a biological system would consist of several hundred neurons, and is a two dimensional array processor. An array of 100 of these elements has been simulated by a computer. Recognition occurs with this array by forming a cross correlation of the input with a stored pattern.

A theoretical model of the coding and processing of visual information in the visual system is described by Rosenberg and Wilkins 1968(40). A simplified model is simulated which has two networks, one of horizontal line detectors and, one of verticle line detectors, which operate on an 18x18 input matrix. After an input pattern has been presented to the model two lists are generated which give a unique representation for any shape, invariant to change in size or position.

An interesting feature extracting system has been simulated by Fukushima 1969(16). This system uses successive layers of analog threshold elements and is applied to the task of character recognition. It generates a set of matrices which correspond to the input matrix and contain extracted features, e.g. line endings, lines with a particular orientation, etc.

1.4.2 Theories of Visual Perception.

The method by which patterns are perceived in man is still not known although there are many theories on this subject. The theories which relate to this project are the ones in which a sequential approach utilising the movements of the eye is considered. One of the earliest of these theories was proposed by Hebb 1949(20) in which perception is realised by 'phase sequence' of activities of 'cell assemblies'. A cell assembly is a collection of cells which when stimulated by an event (e.g. a movement of the eye or the detection of a feature in the visual cortex) can act briefly as a closed system, maintaining this activity after stimulation has ceased. When an object is viewed, a sequence of these cell assemblies some related to eye movements and some to extracted features will be activated. This is referred to as a phase sequence and results in a further cell assembly being activated which indicates the form of the object. Although these cell assemblies have never been detected, Legéndy 1966(30) has suggested that they would have to contain between several hundred and several thousand cells for each assembly.

In a more recent theory by Noton 1970(34) it is proposed that the internal representation of a pattern is a feature network in which the features of a pattern and the shifts of attention,required to pass from feature to feature across the visual field, are recorded. The feature network does not contain all possible attention shifts between features but only those which occur with some frequency. When an attempt to recognise a pattern is made, the recognition system tries to match a feature network with the pattern by executing a sequence of attention shifts specified by the feature network. It is important to note that it is the feature network which directs the matching process. The main difference between this theory and Hebbs is that the feature network is composed of memory traces recording the occurrence of feature detecting and attention activities, whereas the phase sequence is formed by interconnecting the cell assemblies themselves. A more general discussion of the background of this theory is presented by Noton and Stark(35).

The finally developed system differs in many ways from the mechanism of the eye. The most marked difference is that it smoothly tracks around the contours of an object rather than track with rapid saccadic attention shifts which are found with the eye. However, the preprocessing methods found within the retina of the eye and visual cortex and the hypothese concerning the perception mechanism within the brain may still be useful in the development of the proposed system.

1.5 STRUCTURE OF THE THESIS.

The general scheme for the systems which will be described in this thesis is illustrated in Fig. 1.5.1. An optical transducer obtains information from a small area of the visual scene. An adaptive learning network is taught to control the position of attention of the optical trans-

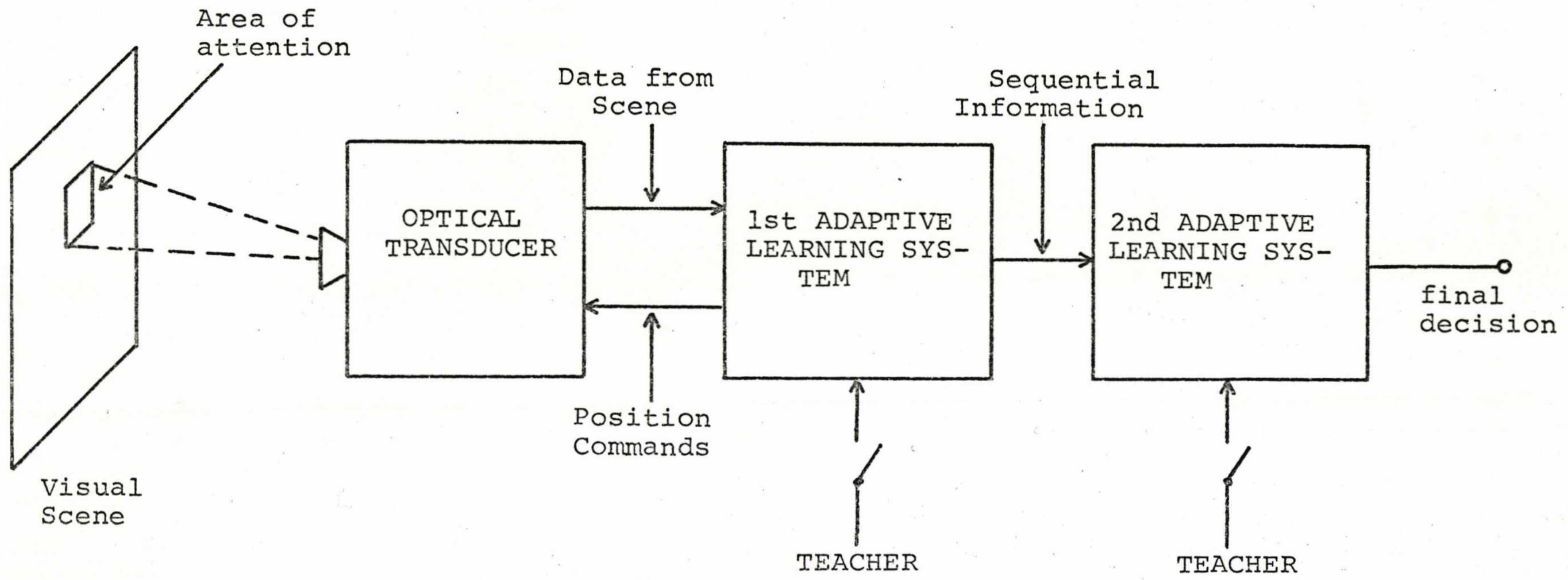

Area of attention
Data from Scene
Sequential Information
OPTICAL TRANSDUCER
1st ADAPTIVE LEARNING SYSTEM
2nd ADAPTIVE LEARNING SYSTEM
final decision
Position Commands
Visual Scene
TEACHER
TEACHER

FIGURE 1.5.1

ducer by reacting to information obtained from the optical transducer. The sequential information obtained by moving the area of attention is then input to the second adaptive learning network which is taught to classify this data.

If the first learning network is taught to track the edges or lines of the input pattern, then the sequential information generated on tracking would be similar to that obtained by tracking systems designed for character recognition. The possibility of teaching the system in this way has been considered in detail.

The optical transducer which has been designed and built for this project is described in Chapter 2. This involved interfacing a television camera to a computer in a somewhat unusual way in that only a small area of the scene of view is considered at a time. One of the most similar systems to this is described by Dinn et al 1970(10). However, this was designed with regards to industrial process control and for conversion of images stored on video magnetic tape and there are many differences in the details.

One of the main differences is in the method used for obtaining averaged data when a small size matrix is used to represent a large area of the scene (i.e. each element of the matrix covers n lines of the television scan). Their system can obtain a 5-bit grey level value for each element of the matrix but only considers one of the n lines of the scan relevant to that element. The system described in this thesis was primarily intended for binary input patterns at one grey level at a time. This allows an averaging method to be used which considers all of the relevant n lines when determining the value of an element of the matrix.

Chapter 2 also contains an introduction to the software system designed to control the television camera.

In Chapter 3 the development of, and the results obtained from, the structure for the first learning network (designed to control the position of the area of attention) is discussed. Here a successful adaptation to tracking strategies was obtained.

In Chapter 4 the development of, and results obtained from, the second learning network (designed to classify the input pattern from the tracking information) is discussed. The results from this section are largely negative: reasons and causes are debated.

Several different learning elements were adapted in the two learning systems and these are described in detail in Chapter 5.

In Chapter 6 a description of the pattern processing software system which has been developed during this project is described and Chapter 7 is the conclusion.

CHAPTER 2

THE DESIGN OF THE HARD AND THE SOFTWARE

In this chapter the design considerations and a description of the operation of the software and hardware systems is presented.

This is intended to supply enough information to clarify references made to the computer system in the following chapters.

Further details of the software system are given in Chapter 6 and circuit details of the hardware system are given in Appendix 1.

2.1 SPECIFICATION OF THE HARDWARE.

2.1.1 General specification.

The device was originally considered as a model of an 'eye' which could 'look' at any part of a scene and receive detailed information about that part of the scene.

To decide how the hardware was to be built for this project the following specifications were defined. A hardware device would view a two-dimensional, planar visual scene and send information in the form of a binary matrix (~16x16 bits) about a part of that scene. The computer must be able to dictate to the device where from the scene and over what area the information is to be taken.

2.1.2 <u>First planned device.</u>

As at any one time only a 16x16 binary matrix of information was required it seemed reasonable to receive this information with a 'retina' in the form of a 16x16 matrix of photodiodes. This retina could easily be coupled to the computer. Unfortunately, at this time, this matrix could only be made 6" square and this incurs mechanical problems. The image could be focused on the retina via a zoom lens and the whole system could be rotated in two dimensions by two geared stepping motors (this motion is similar to that of a real eye), see Fig. 2.1.1. The stepping motors and zoom lens would be controlled by the computer. On closer examination this system presented several difficult problems.

1. The zoom lens should have a ratio of at least 6 to 1 and for a 6" square retina would have to be very expensive.

2. The device would be very bulky and it would be very difficult to make a mechanical system to control the retina quickly and accurately (an accuracy of several minutes of arc is necessary). This could partially be overcome by controlling a mirror in front of the lens instead of the whole lens-retina assembly, see Fig. 2.1.2. This would greatly reduce the mass to be moved but it would also limit the area of the scene which can be viewed.

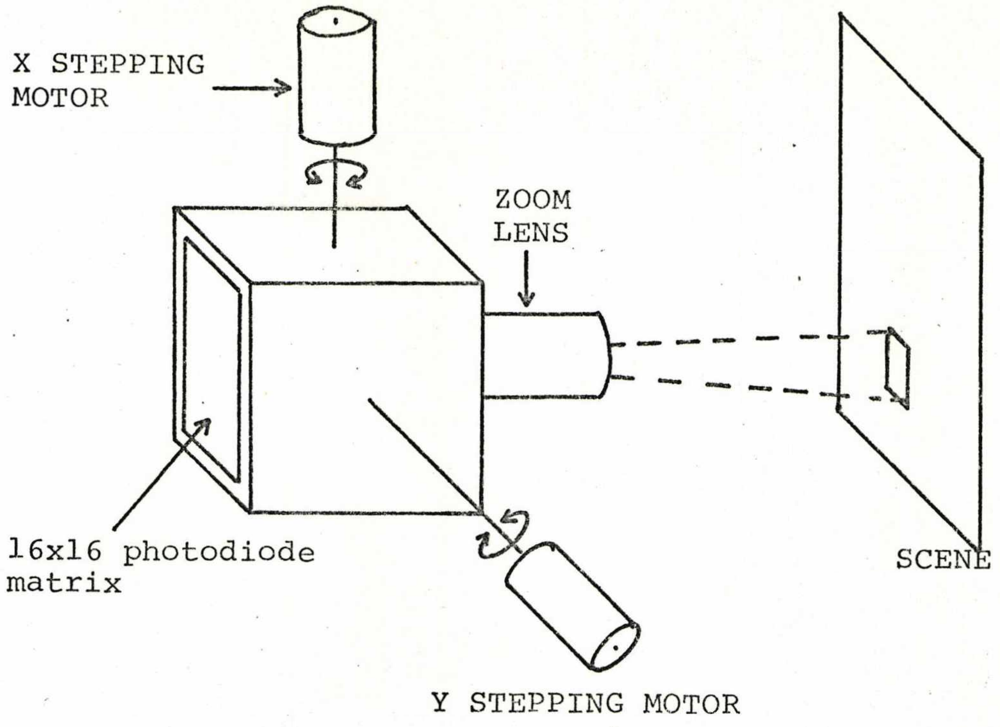


FIGURE 2.1.1

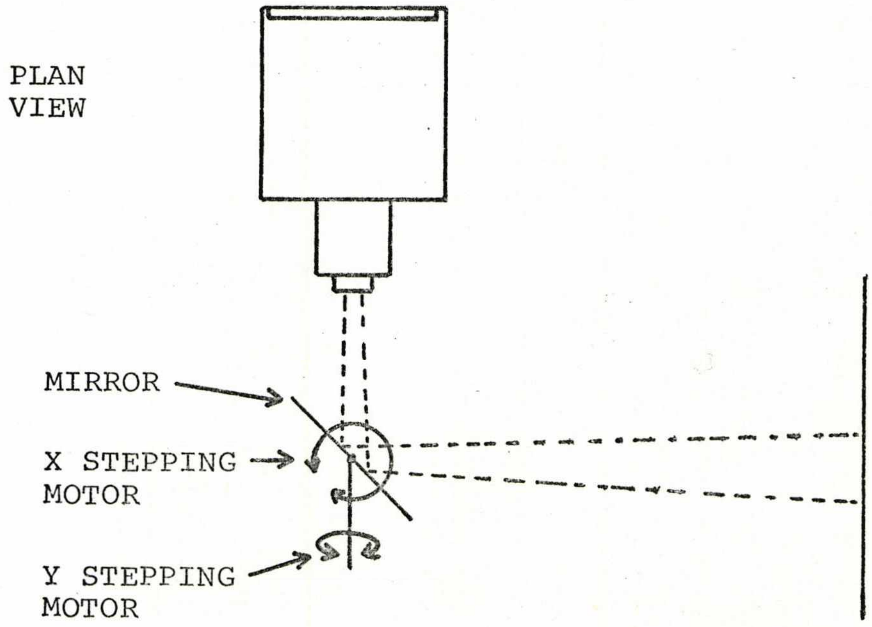
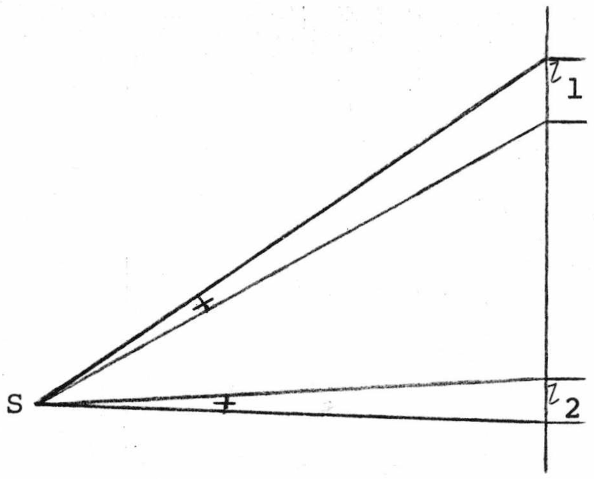


FIGURE 2.1.2

3. If the object is two-dimensional and the device looks at it from a fixed point then the edges become distorted due to the different angle subtended to that point, see Fig. 2.1.3. Using a 16x16 matrix and allowing a'distortion'in the order of one bit a maximum total viewing angle of only 30° is possible.



$l_1 > l_2$, but appears the same when viewed from S.

FIGURE 2.1.3

4. Due to the large size (and low relative efficiency) of the photodiode matrix a large amount of light is required to illuminate the scene. It was estimated that the amount of light required was more than two orders of magnitude greater than was practical.

2.1.3 Second planned device.

Due to the above problems another method of building the device was considered. This method involves a television camera and is shown in Fig. 2.1.4.

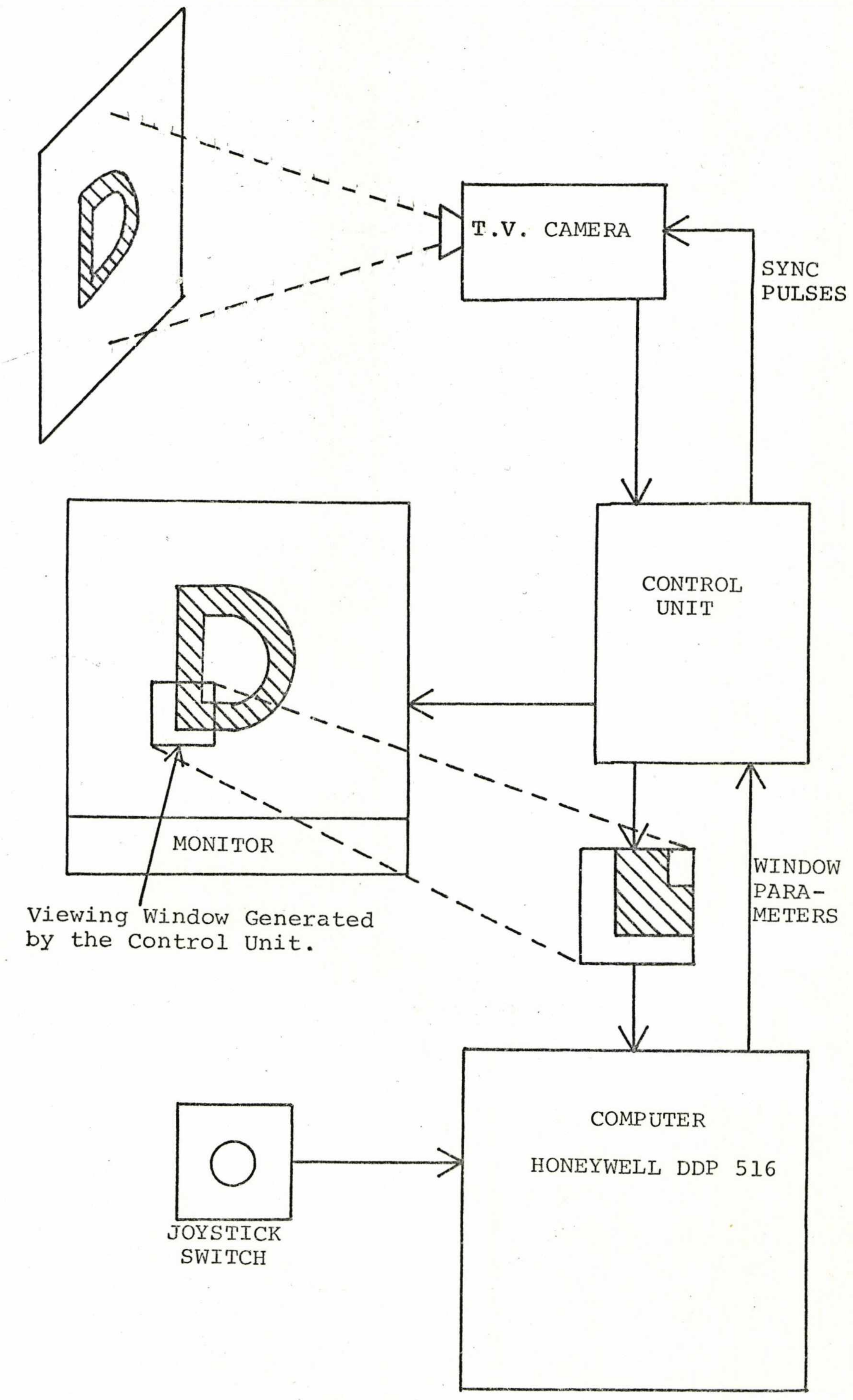

T.V. CAMERA
SYNC PULSES
CONTROL UNIT
MONITOR
Viewing Window Generated by the Control Unit.
WINDOW PARAMETERS
COMPUTER
HONEYWELL DDP 516
JOYSTICK SWITCH

FIGURE 2.1.4

With this system the camera remains stationary and its field of view is determined by its lens. The whole viewing area must be scanned all the time. The control unit then electronically selects information from the video signal only at the part of the scene that is of interest.

The main advantages of this system are as follows:-

1. There is no problem in accuracy, or time lost in changing position of the area of interest as it is now done electronically instead of mechanically.

2. There are no light intensity problems, the camera is designed for normal lighting conditions and has an automatic intensity control which works over a large range.

3. The 'retina' i.e., the camera vidicon surface is plane and stationary and hence the distortion at the edges which occurs when the retina is moved does not exist at all.

4. The camera is a small but strong industrial piece of equipment hence no special care is needed when handling it, and it can easily be serviced.

The main disadvantages are:-

1. A complex control unit is necessary to obtain the information from the camera.

2. The zoom operation can only be done in quantised steps. This is because of the television scanning system, which involves a set number of horizontal lines. The vertical information must be averaged over several lines for each bit hence, for 16 bits, the zoom must average over a multiple of 16 lines.

3. The lens, vidicon system is not very linear in its light response (the video signal is low at the edges of the scene). Hence, it is difficult to sample at one brightness level over the whole scene.

This second system is the one that was actually built.

2.2 THE HARDWARE SYSTEM.

The general layout of the television camera hardware system is shown in Fig. 2.2.1. When the computer wants to obtain a matrix from the system it sends all the settings of the matrix to the control unit (i.e., x and y coordinates, zoom etc.). The control unit then obtains the matrix from the camera and sends it to the computer in 16, 16-bit words.

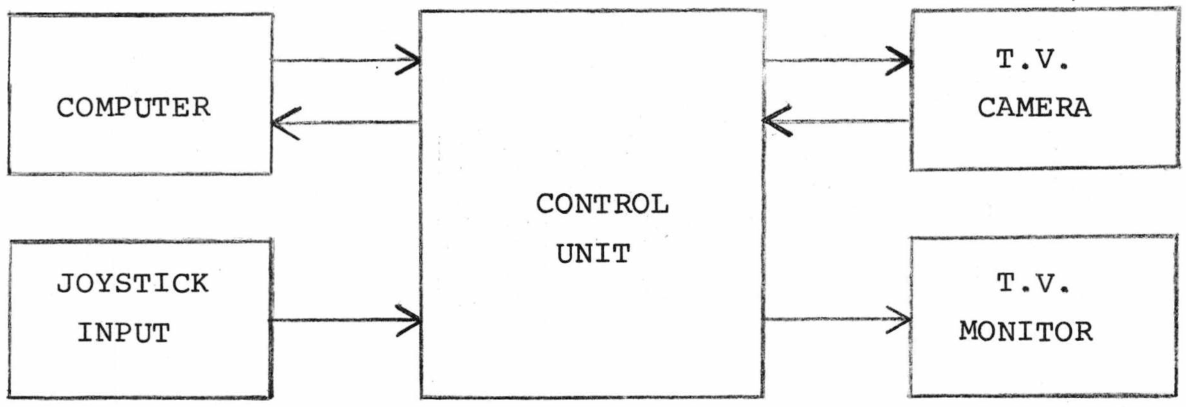


FIGURE 2.2.1

The T.V. monitor shows what the camera is looking at. The area over which the data is being obtained (i.e., the viewing window) can also be displayed on the T.V. screen. The "joystick input" consists of a remote control box with a joystick switch. There is also a two-positon function switch in the box. The outputs from these switches are connected to the computer interface and they have no direct effect on the camera hardware. The joystick input has been built to enable a teacher to control the position of the viewing window. The position of the joystick and the state of the function switch may be detected by the computer program and the effect that these switches have is defined within the program, as described in section 3.2.1.

The control unit contains all the logic for the system. It handles the information from the computer interface, obtains the required information from the camera and sends it back to the interface. It also generates the window displayed on the T.V. screen. The unit is shown in more detail in the block diagram of Fig. 2.2.2.

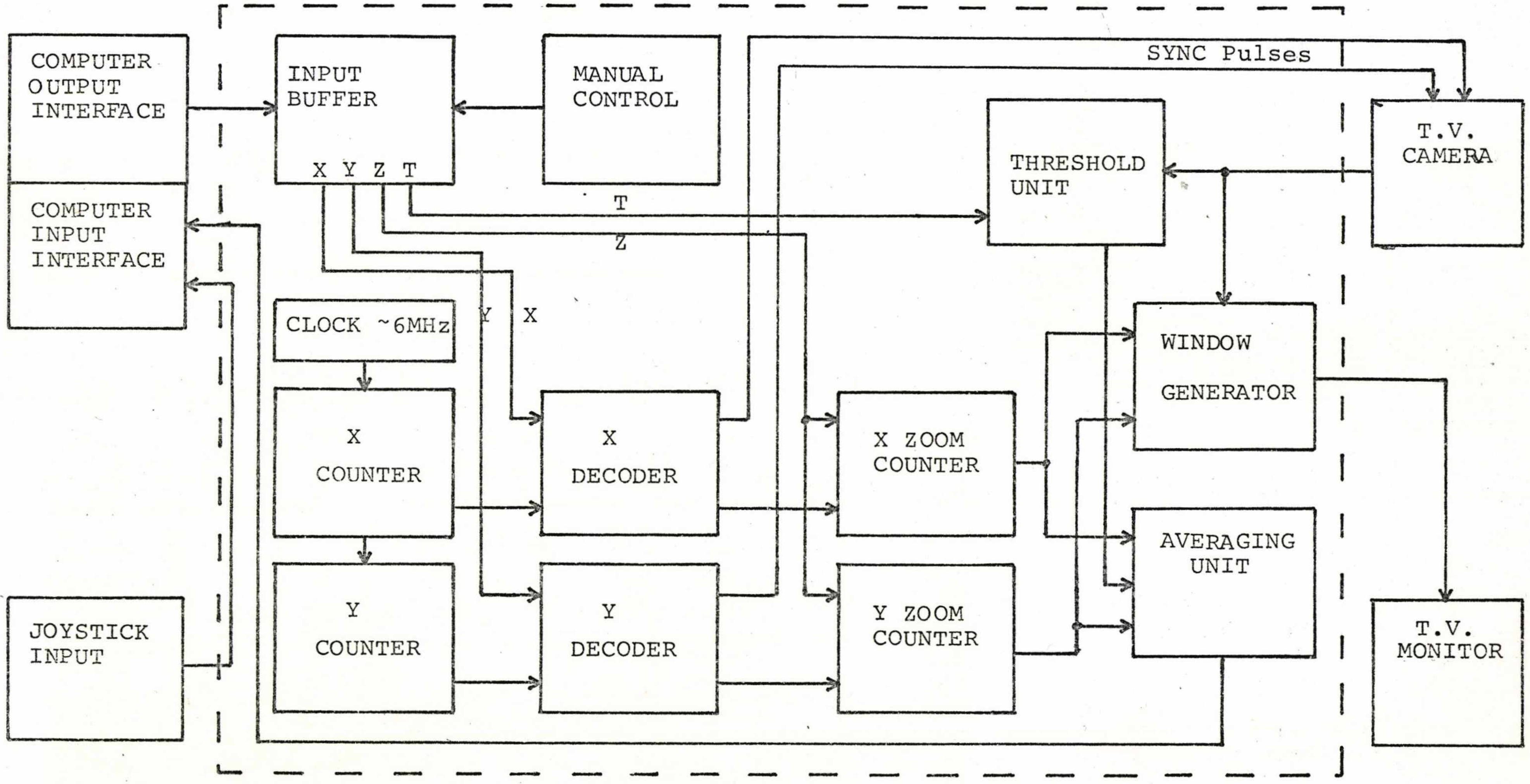

COMPUTER OUTPUT INTERFACE
COMPUTER INPUT INTERFACE
JOYSTICK INPUT
INPUT BUFFER
X Y Z T
MANUAL CONTROL
SYNC Pulses
T
Z
THRESHOLD UNIT
T.V. CAMERA
CLOCK ~6MHz
Y
X
X COUNTER
Y COUNTER
X DECODER
Y DECODER
X ZOOM COUNTER
Y ZOOM COUNTER
WINDOW GENERATOR
AVERAGING UNIT
T.V. MONITOR

FIGURE 2.2.2

The 6 MHz clock controls the main timing of the system. The clock pulses are counted by the x counter, each clock pulse determines the time taken to obtain one bit of information from the camera.

The first 256 increments of the x counter define the time during which data may be obtained from the screen. At a higher count the decoder sends a sync pulse to synchronise the line scan of the camera and then the counter is reset by the decoder for the next line. This is illustrated in Fig. 2.2.3.

Every time the x counter is reset, the y counter is incremented (i.e., each increment of the y counter corresponds to a line of the scan). In the y direction the lines are used as a way of obtaining the quantisation, i.e., 1 line $\equiv$ 1 bit high. Hence, the first 256 increments of the y counter define the time when data may be obtained, at a higher count the y decoder sends a sync pulse to synchronise the frame of the camera with the system. This timing is illustrated in Fig. 2.2.4.

This arrangement allows the area where data may be obtained from the frame to be divided into a 256x256 matrix. Each bit of this matrix may be referenced by the position of the x and y counters.

The position of top left corner of the viewing window is defined by two 8-bit addresses sent via the computer output interface. The first defines the x

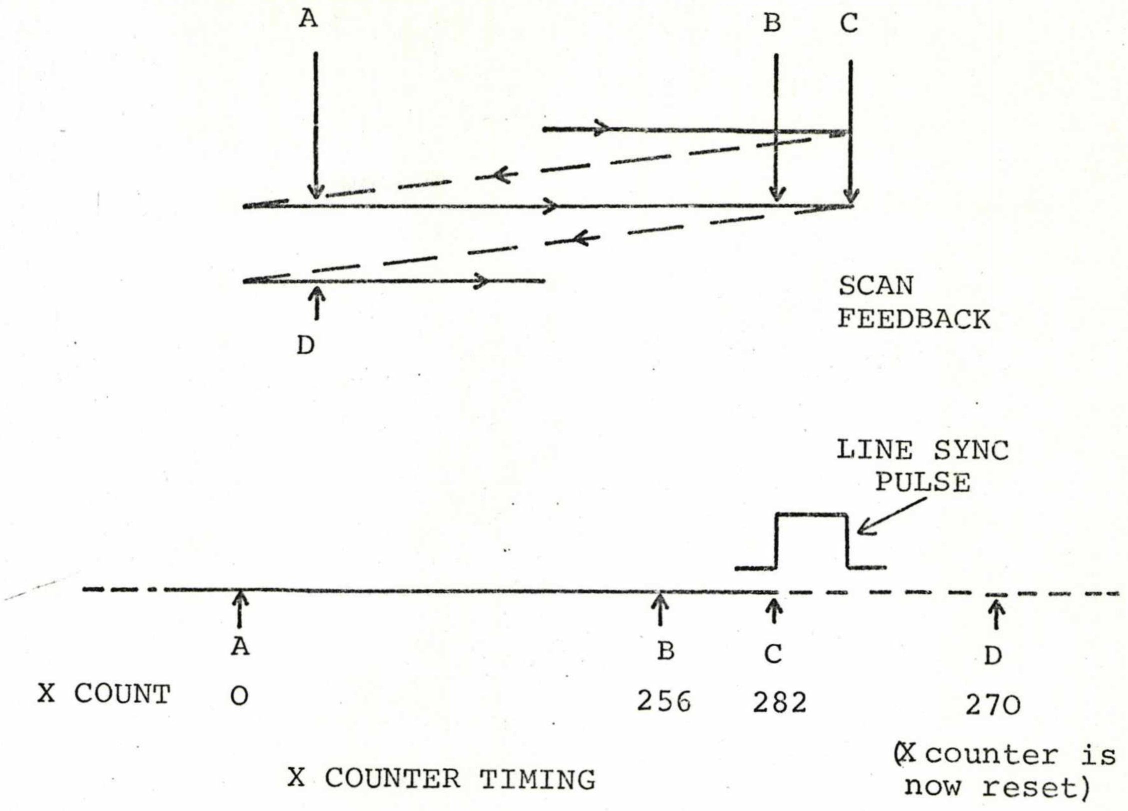


FIGURE 2.2.3

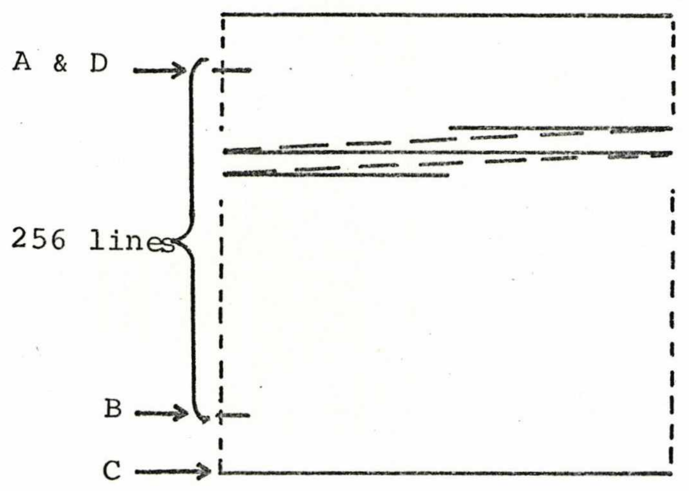
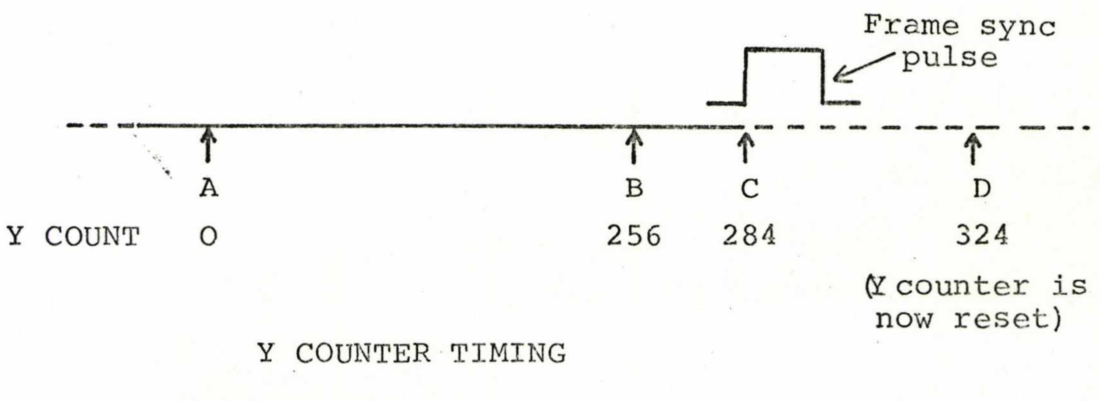


FIGURE 2.2.4



coordinate and the second the y coordinate. These addresses are fed into the x and y decoders respectively (via the lines x and y in Fig. 2.2.2).

Then, when this specified point has been reached by the x and y counters, pulses are sent to the x zoom counter and after every line the y zoom counter is incremented.

The size or magnification (zoom) of the viewing window is sent from the computer output interface to the zoom counters (line z in Fig.2.2.2). Once activated by the decoders, the zoom counters become active for zoom x 16 pulses each scan for the x zoom counter and zoom x 16 lines for the y zoom counter. Hence, the zoom counters are active when the area of the scene defined by the viewing window is being scanned.

The data from the camera is in the form of a composite video signal. This is fed to the threshold unit and to the T.V. window generator. The threshold unit is a comparator which decides which parts of the video signal are "white" or "black" and the resultant digital output is sent to the averaging unit. The voltage level at which the comparator decides between white and black is set by the computer and sent to the threshold unit (via the output interface and the line T on Fig. 2.2.2.)

The averaging unit accepts information from the threshold unit when it receives pulses from the two zoom counters. It generates the output matrix in the form of 16, 16-bit words.

As each 16-bit word is generated it is sent to part of a 256-bit store in the computer interface. When the last 16-bit word has been sent to the interface the control unit indicates to the computer that the data is ready. The computer can then input this data. No more data is sent to the interface until the computer indicates a data request.

The window generator detects when the two zoom counters are active (i.e., when the window is to be displayed) and then adds a d.c. level to the composite signal. The resultant signal is fed to the T.V. monitor and the window is displayed on the monitor as an area of the picture where the brightness level is different to the rest of the picture.

The manual controls (shown in Fig. 2.2.2) are mounted on a panel in the control unit. These switches allow the operator to override any or all of the functions which are carried out by the computer output interface. This is useful mainly when setting up and when working on the equipment off-line from the computer.

It has already been said that the averaging unit under the control of the zoom counters accepts information from the camera, averages it to a 16x16 matrix and sends it to the computer input interface. The way that the unit averages the information is not a strict mathematical average. The unit finds the average of every line of the scan as it is input and then finds the average of these resultants to decide if the bit is a 1 or a 0. For example, consider the case of a zoom of 6 then a 6x6 matrix of the view will contribute to each bit of the output matrix. The way that the unit would handle one of these 6x6 matrices is illustrated in Fig. 2.2.5.

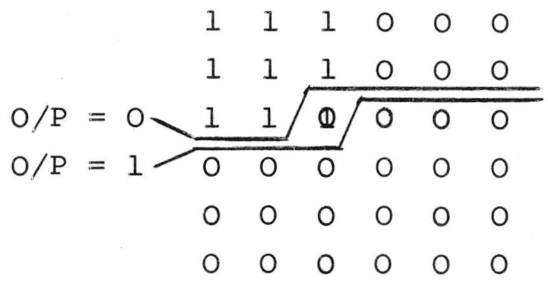
```
1 0 0 0 0 0        0
1 1 0 0 0 0        0
1 1 1 0 0 0        1
1 1 1 0 0 0   =>   1   =>   1
1 1 1 1 0 0        1
1 1 1 1 1 0        1
From Camera                1 bit of output
                              matrix
```

FIGURE 2.2.5 METHOD OF AVERAGING.

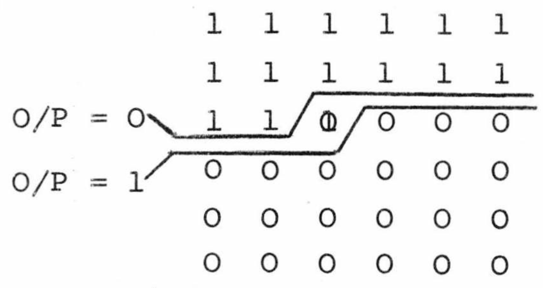
It counts the number of bits in each line and if this is greater than or equal to half the total number of possible bits in a line (i.e., 3 in this case), then that line is represented by a one. This is done for all the lines. The number of lines which are represented by a 1 are then counted and if (applying the same logic as for each line) three or more ones are present then the corresponding bit in the output matrix is made a 1.

For a proper mathematical average all the bits of the 6x6 matrix should have been added together and the resultant compared with half the square of the zoom (i.e., $\frac{Z^2}{2} = 18$ in this case). The first method was used because it is easy to realise in hardware, and has been achieved mainly with 17 4-bit counters. If the proper averaging system had been used the controlling logic would have been a lot more complex and twice as many counters would have been required.

Due to this method, the averaging unit will not output a 1 for patterns with less than $\frac{Z^2}{4}$ bits set as for example:



It will certainly fire if there are more than $\frac{Z^2}{2}$ bits set as for example:



The method is shape dependent but is simple and was found adequate in the experiments.

The number of bits which decides the threshold (i.e., 3 in the case of a zoom of 6) is half the value of the zoom. This number is stored in a counter which automatically counts half the zoom value when the zoom is set. If it is desired to change this threshold, any number (less than 16) may be loaded into this counter by the computer after the zoom has been set. For example, if in the case of the zoom of 6 this threshold was set at 1 (instead of 3), then if any bit of the 6x6 matrix is a 1 the output to the averaged matrix will be a 1. This means that the system is very sensitive to any object but does not give as much detail.

This could be useful in some experiments and the software has been written to enable this to be used. A photograph of the equipment is shown in Fig. 2.3.5.

2.3 THE SOFTWARE SYSTEM.

A software system was written to make the programming of experiments as easy as possible. This is intended for use with learning networks which require pattern manipulation. It contains the programs to control the camera so that it may be used without a knowledge of the hardware details.

2.3.1 Design considerations.

For any experiment concerned with learning networks the following parts of a software system were considered necessary.

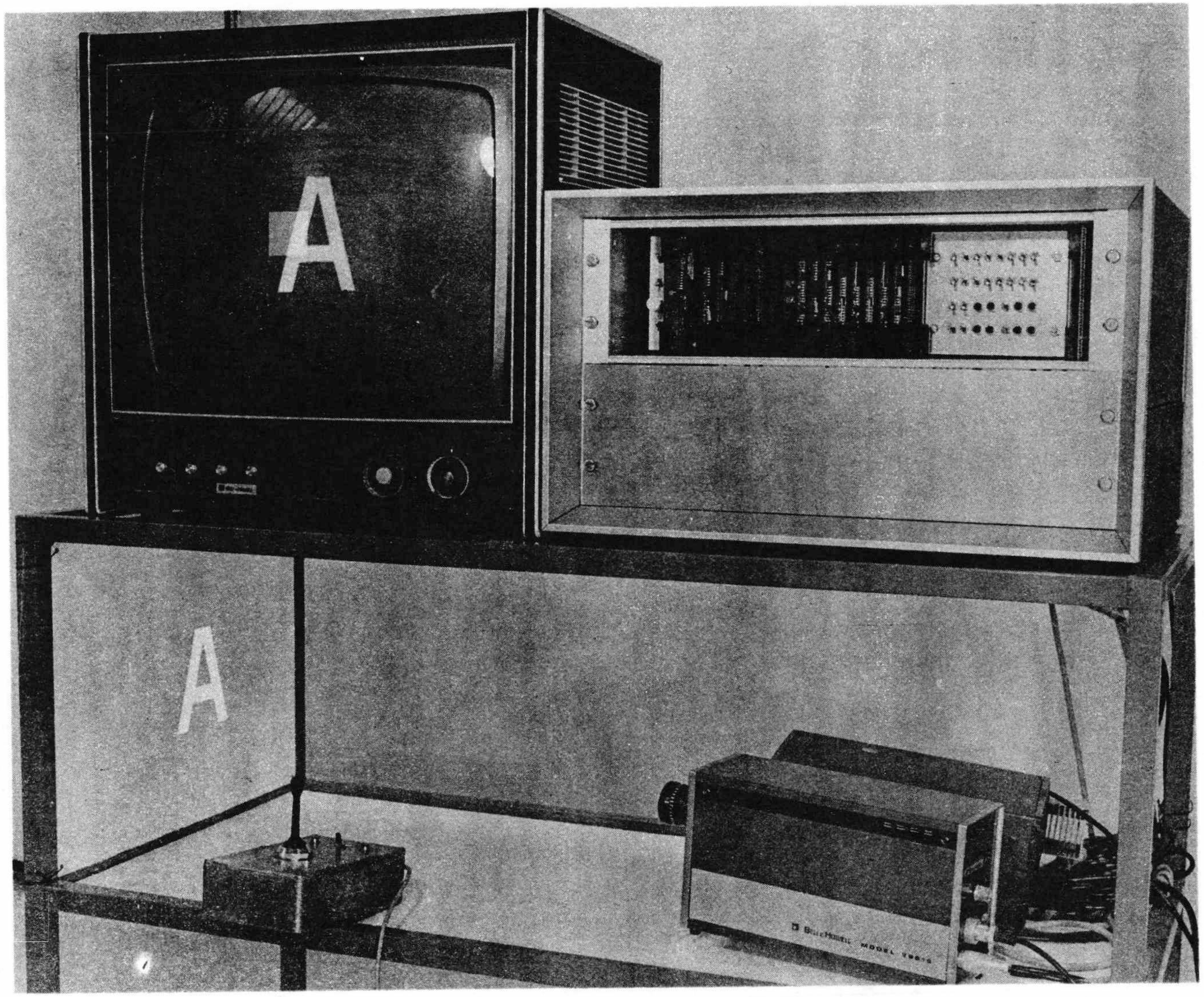


FIGURE 2.3.5

1. A set of executive routines to control all the peripherals including the camera.

2. A library of standard routines concerning pattern manipulation and data organisation.

3. An organised data workspace so that pattern information concerned with an experiment is readily available.

4. An on-line access to enable a user to manipulate manually the data in the workspace.

5. Debugging facilities to enable easy debugging of a new experiment.

In writing this software several constraints had to be considered.

Speed. Most experiments with the camera take place in real time, also most experiments which involve cycling learning nets take a lot of processing time hence, it is very important that the pattern handling routines and simulation routines should operate as quickly as possible.

Space. Space in the computer is very limited. The machine (a Honeywell DDP 516) with a 16-bit word length, had originally 8K words of store and now has 16K words. Further expansion is difficult due to both hardware and software limitations. So all parts of the system must be kept very compact.

Simplicity. This software system is mainly designed for experimental use and users who may wish to understand how parts of the system work or who may even want to change the parts themselves. Hence, standard methods of operations in, for example, the setting up of loops, the calling of subroutines etc. have been established and short cuts for particular cases have been avoided.

2.3.2 The structure of the system.

The software backing system consists of three main modules. These are: an executive module which controls all the peripherals; a main program which sets up the data workspace and allows on-line access and manipulation within it, and a library of standard subroutines which are available to all modules.

The actual experiment is written as a subsystem to the software and acts as a fourth module. There is an optional fifth module which is a debugging system. This can be loaded into the workspace and is overwritten when it is not used.

All modules except the debugging module are loaded in the 'bottom' of the available core and the data workspace is allowed from the end of the last module to the end of the available core. This latter limit can be changed after the program has been loaded if it is desired to use the top of the core store for some other reason e.g., for multiprogramming.

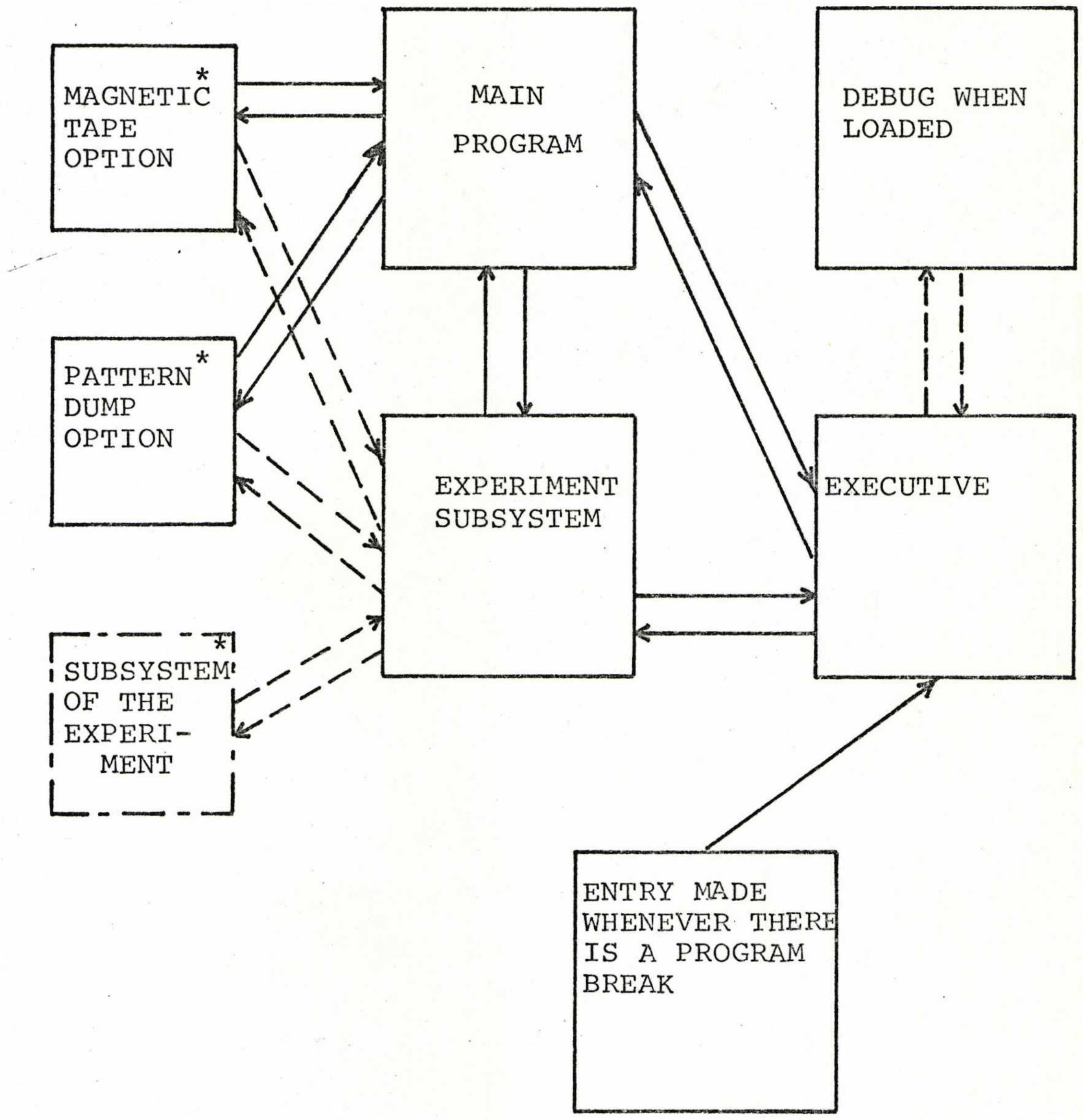
Fig. 2.3.1 shows the interactions between the sections of the software system as they appear to the on-line user. Each section has its own different command status. Usually, unless the experiment is running, the program waits in a command status for a command to be input. Each command status has a unique set of valid commands which it can accept and execute. Commands are input from the command device, this may be any peripheral which inputs characters but it is usually the teletype.

The Executive Input/Output Module.

This module is equivalent to the executive or director in a conventional operating system. It contains all the routines for communicating with the peripherals. Its command status is entered by causing a program break from the command device. In its command status it allows peripherals to be allocated to different channels. There are two exits from this command status, one to the main program and one to the experiment subsystem.

The Main Program.

When starting, the system enters the main program and waits in its related command status. The main program does all the housekeeping for the system which is defined below. Hence, general purpose operations are possible here and do not need to be written into the experiment subsystem. The main program divides the data workspace into blocks of 16 words (16x16 bit binary patterns)



———— Definite links
-------- Links depending on conditions
* Three optional subsystems are drawn, more can be made available

FIGURE 2.3.1

and contains the following on-line functions.

1. Manipulation the workspace e.g., the command 'AN 4 TO 5' will logically AND pattern 4 to pattern 5 and 'OS 10 TO 30' will output on paper tape the patterns 10-30 inclusive.

2. Control of the television camera. A set of commands enables every parameter sent to the camera to be varied and data from the camera to be stored in the work-space.

3. General purpose commands e.g., the command 'PV' enables a heading to be printed onto paper tape. Some commands are also useful for limited debugging e.g., "DC '1000" will display core locations '1000 to '1017 on a visual display and the user can observe how and when they change value.

The Library Module.

This contains general purpose routines used by other modules e.g., the command AN 4 TO 5 given in discussing the main program, this would use the following library subroutines

INCOMMAND	to input the command
INUMBER	to input the two pattern numbers
SORI	to check that these numbers are within range
AND	to do the operation of ANDing the patterns.

Some routines have a command status which enable parameters within them to be set. Examples of these are shown in Fig. 2.3.1.

The Experiment Subsystem.

This is the experiment which is written by the user. Any DAP program can be used as a subsystem however, if the software backing system is considered when writing the subsystem, then it can contain the following features.

1. A command status to enable parameters within the experiment to be varied.

2. All pattern storage and learning network storage can be allocated in the data workspace.

3. All general functions e.g., pattern manipulation etc. can be carried out via the library routines.

In this project several different experiment subsystems have been used. Examples of these occur in section 6.4.

All the learning networks are organised in the experiment subsystem. The operation of the learning networks is assisted by a set of subroutines from the library. Many different kinds of learning elements are made available by different subroutines and only the subroutines which are specifically requested by an experiment subsystem are loaded.

There is a hardware learning machine 'Minerva' which is linked to the computer. An experiment subsystem may use Minerva via some of these library subroutines.

The other learning network subroutines simulate the learning elements within the core store. The active store for simulated learning elements is set up in the data workspace. Hence, the size of the learning networks may be varied and also the main program has easy access to the learning element stores.

The Debug Option.

A special debugging program for the DDP 516 computer has been written at this University.(3) As well as a general debugging aid it also allows small DAP-like programs to be input on-line and modifications to the existing program. As a large amount of time is spent debugging the experiment subsystem, it is important that the system should have some debugging aids. It does have some in the main program but these have been limited due to the space they require. The debug program has been written so that it lies within two sectors (i.e., 64 16x16 patterns). It was originally self-contained but now a modified version has been made which can link with the system through the Executive module.

The Debug option may be loaded at any time. It loads into the top of store and takes the space of the top 64 data patterns. It has a command status which can be entered from any command status in the system.

2.3.3 Summary of the software system.

A software system has been written which is to aid experiments using patterns with particular regard to the camera peripheral and SLAM learning networks. It is self-protected i.e., it cannot corrupt itself unless there is a mistake in the experiment subsystem (or if a mistake is made in the use of Debug).

It means that a user can write an experiment subsystem which can have access to all the routines of the extensive library of the system and when necessary can be debugged or changed by a high level debugging system. The resulting program should run almost as fast as is possible and hence be suitable for real time applications.

This system is now being used for another project concerned with learning networks for pattern recognition.(51) This project does not use the camera at all except for occasional data preparations. Many features of the system have been left out of this description but they are described in Chapter 6 which is devoted to the software system.

CHAPTER 3

THE TRACKING SYSTEM

In this chapter the features of a tracking system are discussed. The tracking of thick lines or line drawings is considered. The tracking system is taught by a human teacher and is designed so that it will mimic the operator's actions on the basis of the subpatterns extracted from the viewed pattern. It is hoped that, due to the adaptive learning networks, the system will be able to track patterns not seen during the period of 'training', that is, it should be able to generalise.

Edge following, the simplest form of tracking, is considered first and the tracking system to do this has the following features:

1. It contains learning networks which are taught (by a human operator using a joystick) to make the decisions regarding the movement of the points of attention.

2. It receives information from the patterns only from the local area around the point of its attention, i.e., the point it has at present reached on the pattern due to its tracking.

3. It uses the camera system (this is described in section 3.1).

The general scheme for this system is illustrated in Fig. 3.1.1. This scheme has been further developed to track line drawings. If every line of a pattern is tracked then there should be sufficient information from the tracking motions to recognise the pattern. Also, different organisations of the learning elements within the learning networks have been investigated.

3.1 THE CAMERA SYSTEM.

The camera system, consisting of the camera hardware and its software backing can be represented by the diagram in Fig. 3.1.2. The details of the operation

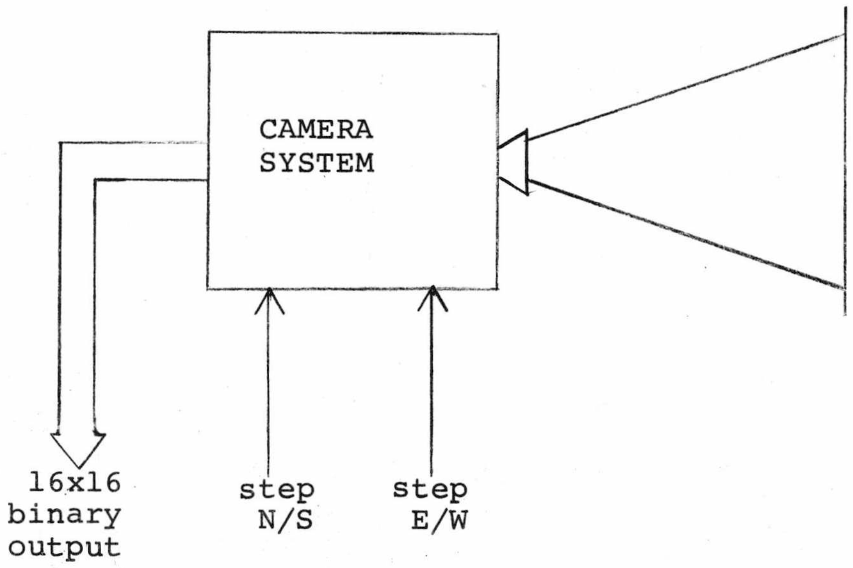


FIGURE 3.1.2

of the hardware are given in section 2.2. The camera light threshold is preset to detect the pattern and the

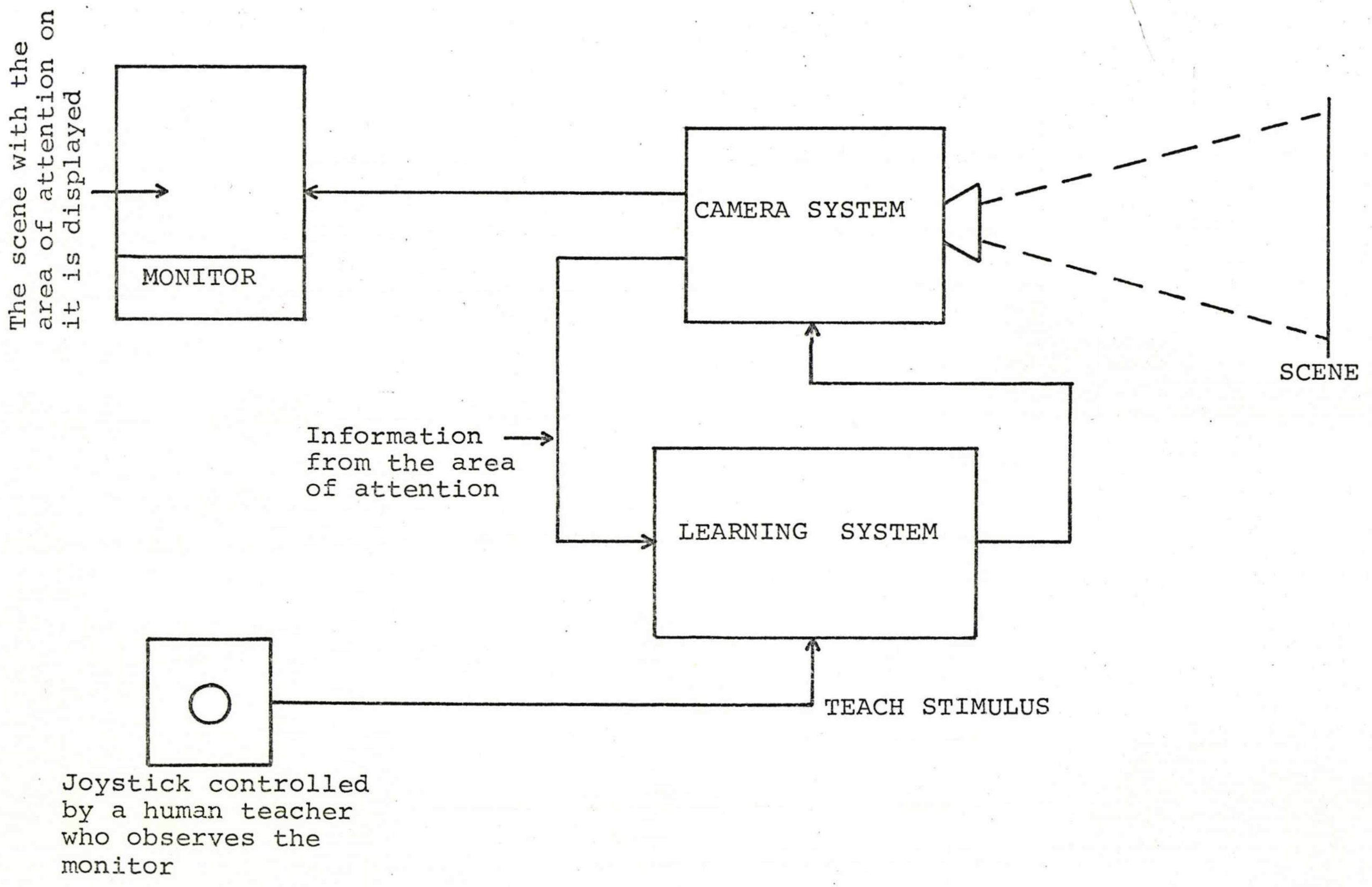

The scene with the area of attention on it is displayed
MONITOR
CAMERA SYSTEM
SCENE
Information from the area of attention
LEARNING SYSTEM
TEACH STIMULUS
Joystick controlled by a human teacher who observes the monitor

FIGURE 3.1.1

zoom value is preset to a suitable size with respect to the pattern size. Therefore, only the position of the viewing window is to be controlled. At each instant of clocked time, the position of the viewing window may be moved in the North/South and East/West directions by a preset number of picture elements. The number of elements moved is called the step size and it is set within the tracking experiment.

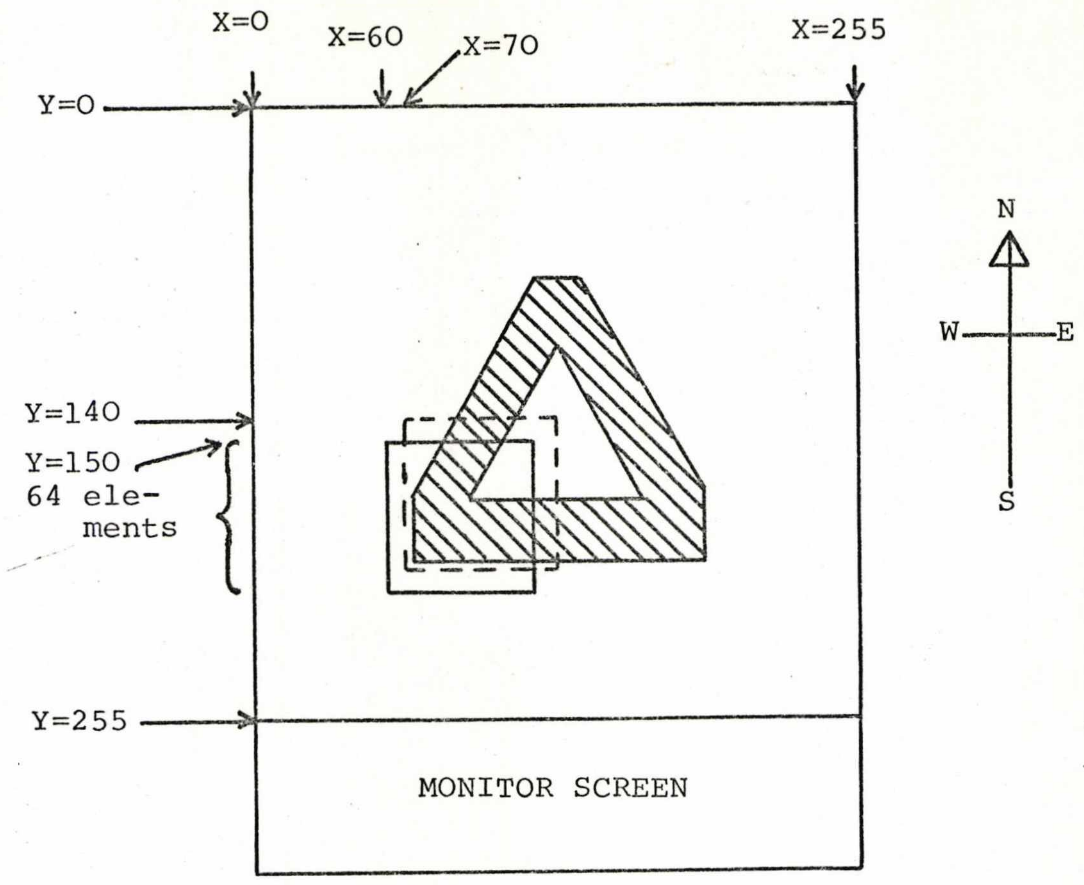
Hence, there are two inputs to the camera system. One to indicate a step in the North or South direction and the other to indicate a step in the East or West direction. Strictly, these inputs are three-level inputs in that no movement at all may be commanded.

The output from the camera is a 16x16 binary representation of the area of the scene within the viewing window. An example of the information obtained from the pattern is shown in Fig. 3.1.3. The new position of the viewing window after the command NE is given is shown by the broken lines.

3.2 THE TRACKING SYSTEM.

3.2.1 Basic structure of the system.

The structure of the tracking system is partially defined by the camera system. The only information from the scene to the tracking system is through the viewing window, and the system must generate the N/S and E/W



Parameter settings

X value = 60
Y value = 140
Z value = 4

Threshold (depends on aperture, set to detect the pattern)

```
1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1
1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1
1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1
1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1
1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1
1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

FIGURE 3.1.3

commands to move the area of attention (i.e., the position of the viewing window). Hence, two outputs are needed from the tracking system (N/S and E/W) and the learning networks have been split into two modules to provide them. This is shown in Fig. 3.2.1. These modules are taught different information but are identical to each other in structure. Each module, whatever its internal structure has the following characteristics:

1. It contains a learning network which receives information from a binary input matrix and which can be taught via the teach input to associate either a high or low output response with this input matrix (i.e., it must be able to produce the same response when this input is again presented to the module without any teach stimulus).

2. A ternary (e.g., N, S and no move) output decision system to drive the camera system.

3. A ternary 'teach' input decision system so that the module may be taught to output desired responses.

In practice, for convenience of the hardware, ternary teach inputs are realised by two binary inputs in which only 3 of the 4 possible binary combinations are permitted.

The ternary output is to match into the ternary input of the camera system, however, if the step size is small then a binary output may be sufficient that is, the

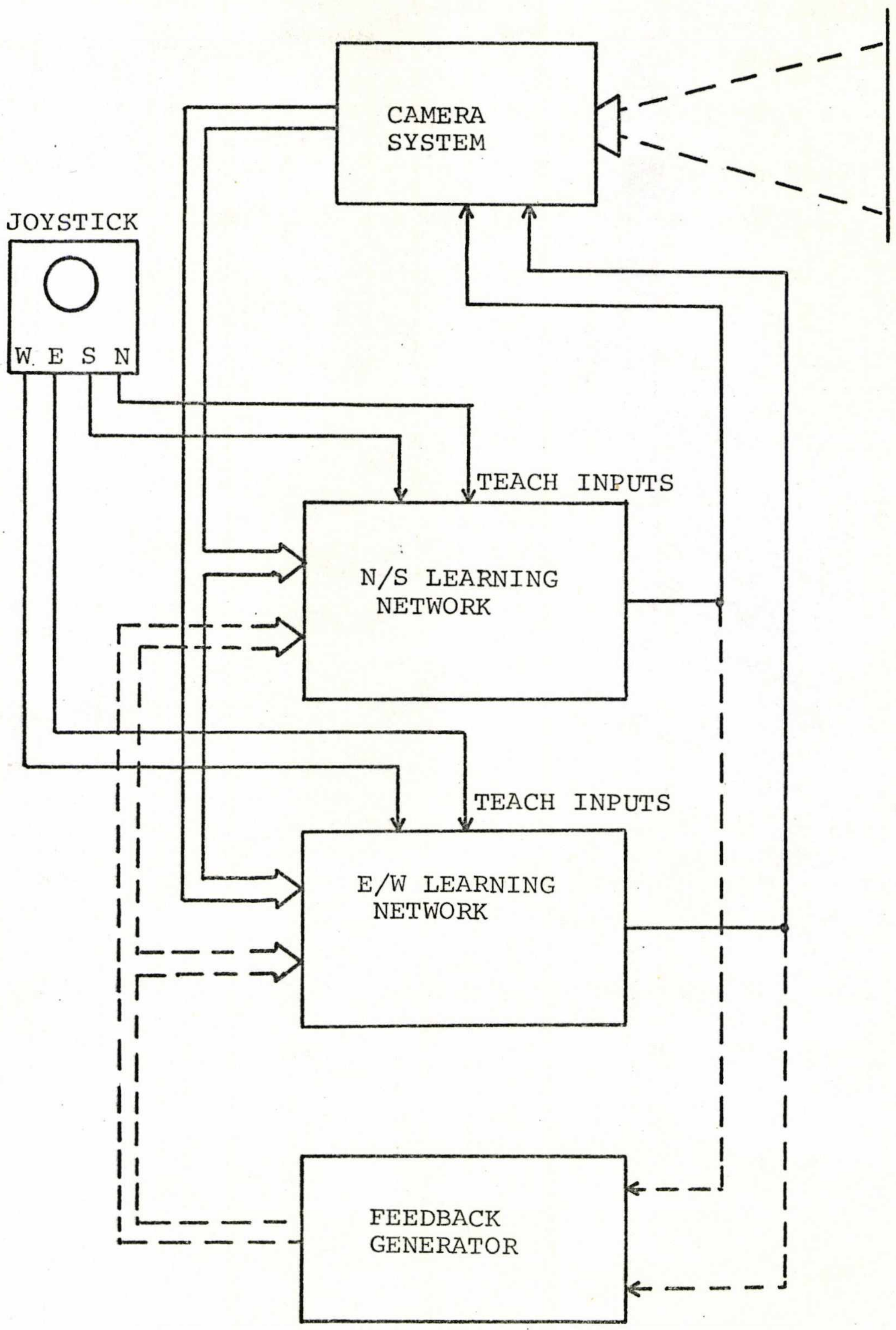

CAMERA SYSTEM
JOYSTICK
W E S N
TEACH INPUTS
N/S LEARNING NETWORK
TEACH INPUTS
E/W LEARNING NETWORK
FEEDBACK GENERATOR

FIGURE 3.2.1

need for a dead zone is removed. This would compel the viewing window to move in the N or S direction with every instant of clocked time i.e., one of the four possible moves (NE, SE, NW and SW) is forced.

Teaching is achieved by a human teacher using the joystick switch. This has a center return and can be pushed in the desired direction of motion. It has four binary outputs, one for each of the four cardinal directions N.S.E.W. and there is an overlap between adjacent directions. For example, it is possible to indicate NE when both N and E outputs will be true. This switch is connected to the teach terminals of the learning modules. When the joystick is pushed forward the N output becomes true and teaches the N/S learning network to give a high response to the input pattern, this results in the viewing window moving N. When the joystick is in the center position, both the teach terminals are false, no teaching takes place and the learning network makes its own decisions. Hence, the three states of the teaching inputs are teach high, no teach and teach low.

3.2.2 Last step feedback.

With the learning module described in the previous section it is possible to associate an input pattern with a command for the direction of the next step.

Consider the system shown in Fig. 3.2.1 (without the feedback) and a particular tracking task, of tracking around a square in a clockwise direction, as shown in Fig. 3.2.2. Now consider the viewing window in the two positions shown, at the top and bottom of the square. At the top of the square the learning networks must associate the pattern in the viewing window with the direction E and at the bottom with W. These two patterns are completely different, so this should be possible.

One can see that a unique direction could be associated with every possible pattern which appears in the viewing window when it is on the edge of the square. This is true for all solid patterns.

However, if we try to use this system for tracking a line drawing of a square,instead of a solid square, an ambiguity arises.

Fig. 3.2.3 shows a line drawing of a square in which the task of tracking clockwise is again examined.

Consider the viewing window in the two positions indicated. It can be seen that the pattern in the viewing window is identical at these positions yet we want to move in opposite directions. Hence, it is not possible to track the pattern using the system in this way.

One possible solution is to displace the position of the viewing window with respect to its position on the line as shown in Fig. 3.2.4. Hence, the viewing window not only detects the line but its position with respect to

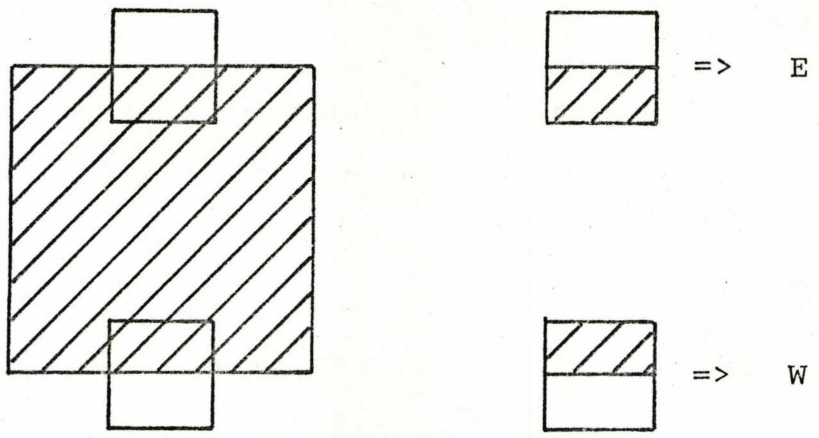


FIGURE 3.2.2

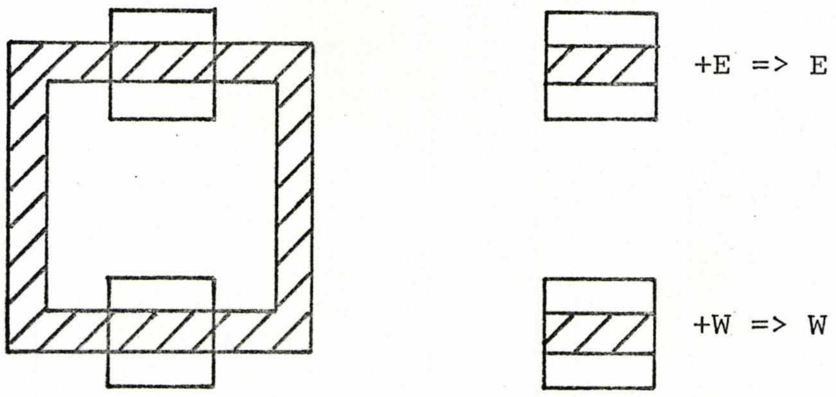


FIGURE 3.2.3

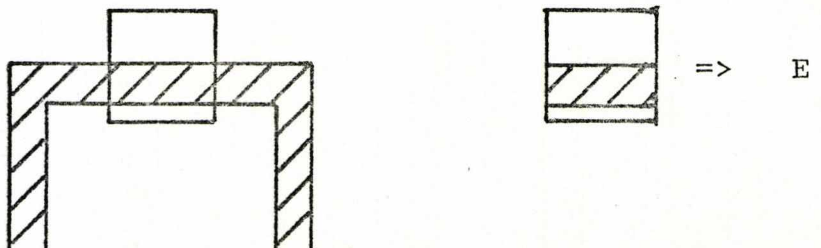


FIGURE 3.2.4

that line determines the direction of motion. The pattern in the viewing window is different at the top and bottom of the square hence tracking is possible. This method has two disadvantages which are as follows:

1. The teacher must be very careful, when training, that the line does not drift much from the desired position in viewing window else the teaching will be incorrect; for different positions in the viewing window mean more than just a directing error.

2. The teacher must plan a strategy so that all the directions associated with particular lines and positions of the viewing window are unique. This is simple with patterns like squares and circles but becomes very involved when more complex patterns with many lines and line junctions are considered.

An alternative solution which has been investigated is to include a memory within the system to indicate the direction of motion in ambiguous situations. For the case of the square this memory need only be one step long and it can be achieved by adding internal feedback around the learning modules. This is called 'last step' feedback.

The way that this feedback is added to the model is shown by the broken lines in Fig. 3.2.1. Last step feedback means that the last direction that the viewing window moved is fed back to the input of the learning network. Hence, the input is now a combination of the pattern in the viewing window and the last direction moved. (Several methods of combining this information have been investigated and the details of these are given in section 3.3.4.) How this feedback solves the problem is illustrated in Fig. 3.2.3. At the top of the square the pattern in the viewing window plus the fact that the last direction moved was W implies that the new direction is W and vice-versa at the bottom of the square. With this system it is possible to track simple line drawings and examples of these are given in Fig. 3.2.5. When using this type of feedback the task of the teacher has been reduced to guiding the viewing window along the lines of the line drawing in the desired direction.

If we limit the function of the viewing window to following a line we can more clearly define what we mean by 'tracking a line drawing'.

Tracking a line drawing occurs when the viewing window traces a circuit over a line drawing in such a way that every line is at some time covered by the viewing window. Covered in this sense means that every part of the line is, at some time, contained within the viewing window.

The tracking is'good' if the centre of the viewing window accurately follows the center of the lines of the drawing and hence the shape of the line drawing can be accurately reconstructed from the motions of the

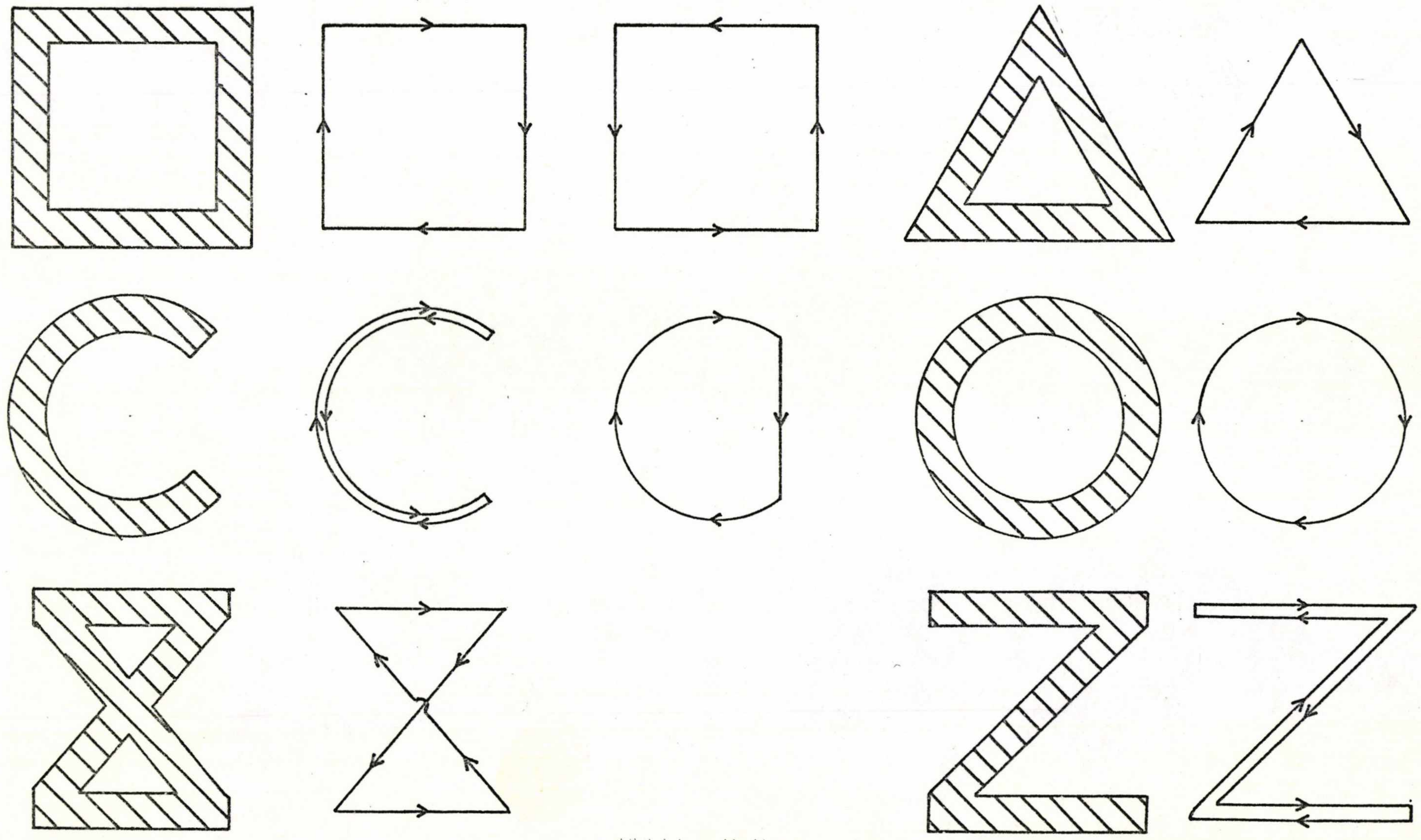


FIGURE 3.2.5

viewing window.

For the edge following task in which the contour of a solid pattern is traced 'tracking the edge' of a pattern may also be defined in a similar way.

Tracking the edge of a pattern occurs when the viewing window traces a circuit over a solid pattern in such a way that during this circuit the viewing window passes over every part of the edge of the pattern.

The tracking of the edge is 'good' if the centre of the viewing window accurately follows the edge of the solid pattern and hence, the shape of the solid pattern can be reconstructed from the motion of the viewing window.

3.2.3 Delayed and damped feedback.

If we wish to track more complex patterns which have joining and intersecting lines, then a further problem appears which is demonstrated in the following example.

This example is illustrated in Fig. 3.2.6 where the pattern and the desired tracking path is shown. The problem occurs at the junction of the center line with the left hand upright line. This is shown in greater detail in Fig. 3.2.7. Table 3.2.1 indicates the information presented to the learning networks with different modes of feedback and the desired response from the learning network

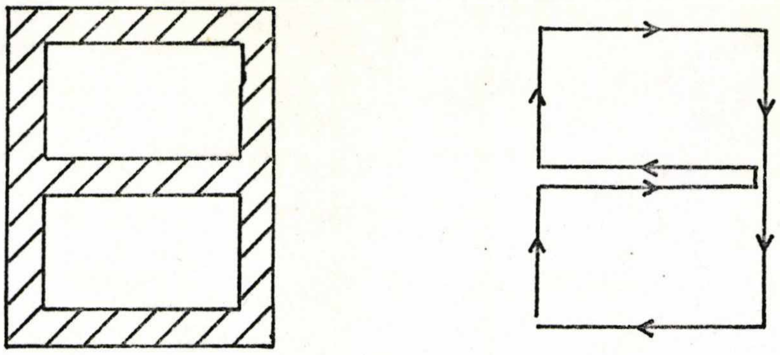


FIGURE 3.2.6

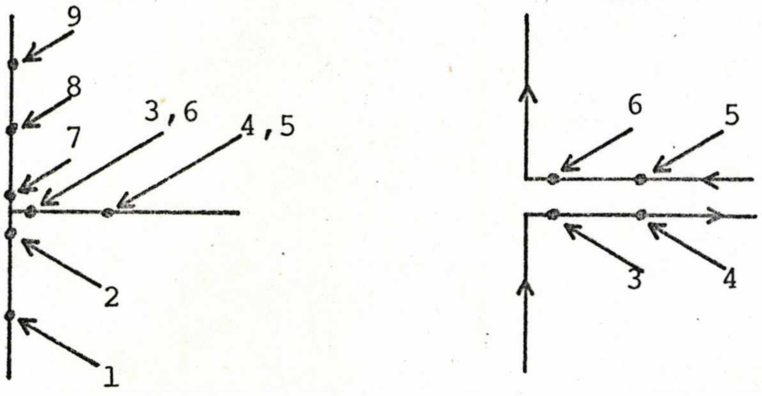


FIGURE 3.2.7

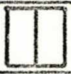
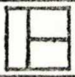
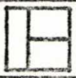
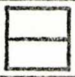
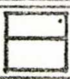
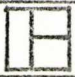
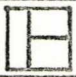
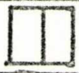
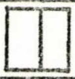
POSITION	PATTERN IN VW	LAST STEP FEEDBACK	DELAYED FEEDBACK	DAMPED FEED-BACK	NEW DESIRED DIRECTION
1		↑	↑	↑	↑
2		↑	↑	↑	→
3		→	↑	↗	→
4		→	→	↗	→
5		←	←	←	←
6		←	←	←	↑
7		↑	←	↖	↑
8		↑	←	↖	↑
9		↑	↑	↑	↑

TABLE 3.2.1

for the different positions shown in Fig. 3.2.7.

Now consider the viewing window in the position (2) just before the first turn is made and position (7) just after the second turn has been made. In these two positions two different outputs are required. However, the last step feedback is identical and the patterns in the viewing window can be very similar if not identical. Therefore, the tracking system with last step feedback only is not able to track the desired path.

This problem arises at the critical point just at the junction. The viewing window changes direction, and hence, there are several values for the last feedback. To overcome this some memory must be added to the system so that information about the direction that the viewing window approached the junction is presented to the learning network until the viewing window has moved away from the junction and the critical point is passed.

Two methods of achieving this memory are discussed here. The first is to delay the feedback by n steps where n is a preset value. This could be achieved by a shift register, for each direction, of length n in which the output from the learning network is input, and the output from it is combined with the viewing window information and presented to the input of the learning network. Hence, any change in feedback will occur n steps after the junction. The directions that this feedback produces are also shown in Table 3.2.1. From the table it can be seen, for example, that between positions (8) and (9) the feedback changes value. This change was initiated by the second turn at the junction. It can be seen from the table that no anomalies like the one for the last step feedback occur for the delayed feedback and hence, tracking is possible with this method.

The second method, damped feedback, is to limit the rate of change of the direction fed back. Hence, although the direction fed back starts to change at the critical point in the junction, it does not complete the change until several steps later. The directions that would be fed back for the example using this type of feedback are also given in table 3.2.1, and again, it can be seen that no ambiguities arise and tracking is possible.

This type of feedback has been investigated and the results are given in section 3.3.5. The feedback vector has been generated in two separate parts, a N/S component and an E/W, which are calculated separately.

The N/S vector for example, may vary between +1 indicating N and -1 indicating S. The simplest way to implement the rate of change of feedback would be to limit the amount by which the feedback vector is changed each step, until one of the maximum limits is reached. This is expressed by:

$$NSV = +1 \text{ for } NV \geqslant +1$$
$$= NV \text{ for } -1 < NV < +1$$
$$= -1 \text{ for } NV \leqslant -1$$

where $NV = NSV' + K \cdot LD$

and NSV is the N/S vector

NSV' is the previous value of NSV

LD is the last direction that the N/S learning module outputs, LD=+1 or 0 or -1

K is a constant which determines the amount of change of NSV per step.

However, this simple method is not sufficient. This is illustrated by an example in Fig. 3.2.8. In this diagram the tracking path is shown in (a) the variation of the last step feedback (LD) is shown in (b) and the variation of the N/S vector is shown in (c) for delayed feedback and (d) for this simple damped feedback. When no direction commands are given then this feedback remains constant at its last value. Hence, at position (7) this feedback is still indicating an N direction.

This has been overcome by allowing the vector to zero if no commands are given. This has been achieved by calculating the vector with the following equation:

$$NSV = NSV' . \underline{(1-K)} + K.LD \quad , \quad 0<K\leqslant 1 .$$

The way that this would work is shown in Fig. 3.2.8(e). This method is equivalent to taking a weighted sum of the past directions where each weight is

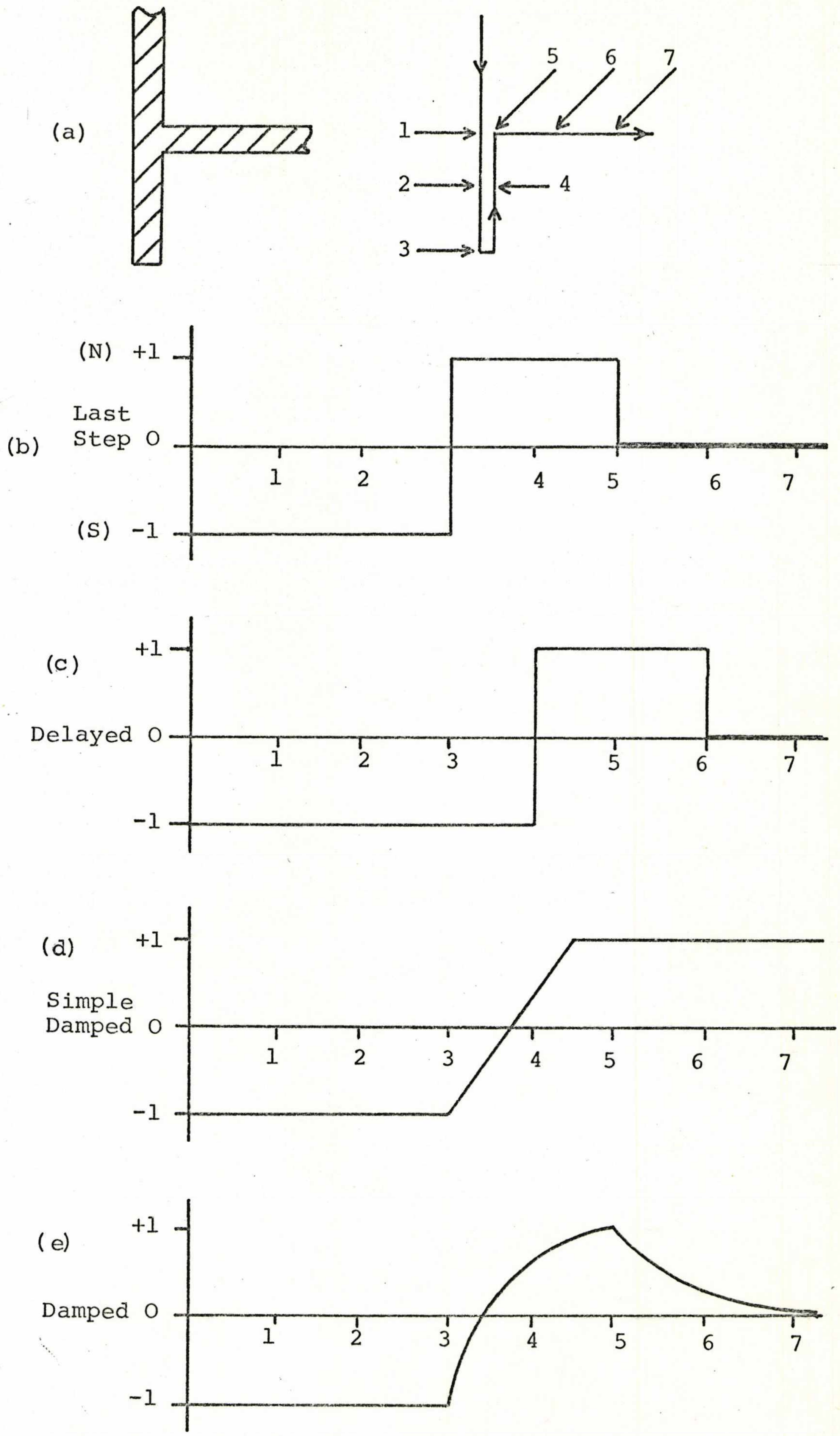

(a)
1
2
3
4
5
6
7
(b)
(N) +1
Last Step 0
(S) -1
1 2 4 5 6 7
(c)
+1
Delayed 0
-1
1 2 3 5 6 7
(d)
+1
Simple Damped 0
-1
1 2 3 4 5 6 7
(e)
+1
Damped 0
-1
1 2 3 4 5 6 7

FIGURE 3.2.8

inversely related to the number of steps that a direction is in the past. Both delayed feedback and damped feedback are affected by the step size. There are usually in the order of 4-8 steps for the viewing window in moving between position (1) and (3) shown in Fig. 3.2.7. For good tracking the step size must be kept small to ensure that the features of the pattern are centred in the viewing window. However, the speed of tracking is proportional to the number of steps taken and is therefore inversely proportional to the step size. The dependancy of the feedback on the step size means that the size of the pattern to be tracked is now relevant. In fact, with delayed feedback there is a definite problem when the critical points of two features occur in the order of n steps apart.

The delayed feedback has an inherent limitation in that it can only feedback one of eight unique values at the critical point of a junction. If one considers the more usual case where the output of the learning module is binary instead of ternary, then only four directions are possible. Also, often only 2 of these directions are valid. For example, if the system tracks a horizontal line well in an E direction then the sequence of commands would be NE, SE, NE, SE ... The N/S value would oscillate in this case and this N/S feedback when delayed is of no use. Hence, there is a definite restriction to the number of different lines which a system with delayed feedback can enter a junction and uniquely remember the direction of entry. With damped feedback this fundamental limit does not exist. The number of different entry lines which may be remembered with this feedback is determined by the number of feedback connections and the size of the learning network.

The damped feedback method is the one which has been investigated and details of this are given in section 3.3.6.

3.2.4 Theoretical Performance.

The best possible performance of the tracking model with respect to the complexity of the input pattern may be determined.

Using damped feedback the model has the following three properties:

1. It can be taught to follow any single line in any direction.
2. When it enters a junction of several lines it can be taught to exit the junction by any one of the lines, even by the one by which it entered.

The above implies that if ever the system proceeds the same way along a line, it has previously tracked, then it is in a permanent loop and has completed a cycle of it.

3. If two or more junctions of a pattern are identical (i.e., the have the same number

of lines joining at the same angles) then the second property concerns them all together as the tracking system cannot distinguish between them.

These above three properties determine the tracking abilities. With such a system it is not possible to teach it any scan path around a pattern. This is illustrated in Fig. 3.2.9. It cannot be taught the first tracking path because this involves making a different decision when entering the same junction from the same direction, as shown in the diagram at the points indicated, and this contradicts the second property. However, this pattern can be tracked by the second path hence, a tracking path can often be found if it is thought out beforehand.

The reason for this limitation is that there is no long term memory in the system to inform it where it has been. There is only a short term memory for enabling the system to pass through features. Further development could be aimed at providing a long term memory via a second feedback loop. If this was successfully added then the tracking system would be able to track any line drawing.

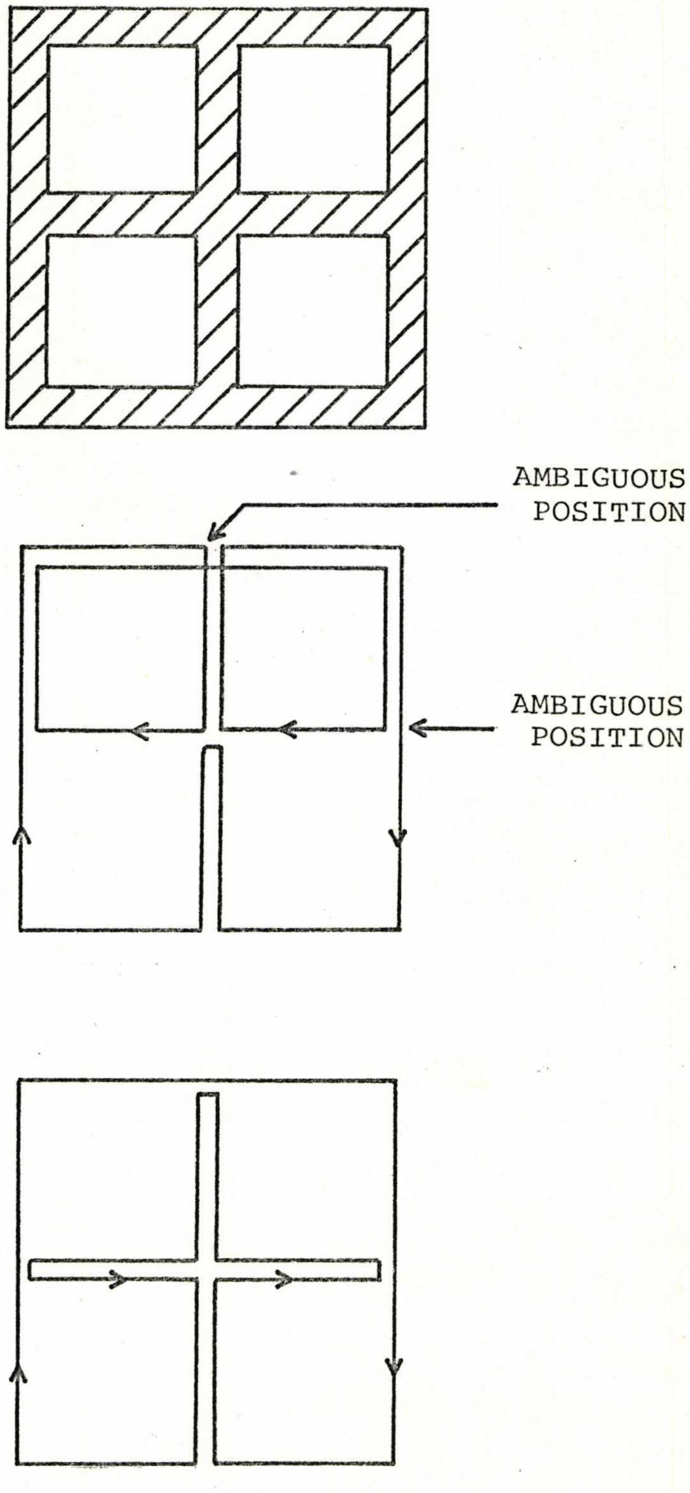


FIGURE 3.2.9

3.3 A Maximum Response Learning System.

3.3.1 Structure of the learning modules.

Throughout these experiments the basic element which has been used in the learning networks is a SLAM. This is described in detail in section 5.1.1.

The structure of the learning elements to form a learning module is shown in Fig. 3.3.1. The learning networks are made of SLAM-16 elements in which all of the teach-clock inputs have been connected together also all the teach-sense inputs are set a 1. Therefore, when the learning network is clocked all of the SLAMs will output a 1 for the particular input pattern.

The clock terminal of the N net is connected to the N teach terminal and the clock terminal of the S net to the S teach terminal.

Initially, the stores of the SLAMs are set at 0. When the joystick is pushed in the N direction then the N learning network will output all 1s for that input. This response, (the sum of all the SLAMs outputting a 1) is compared with the response from the S learning network and the highest response determines whether the output from the module is high or low. A threshold can be set at the output which the difference between the responses must exceed for a non-zero output, otherwise the ternary output will be at the 0 neutral state. In practice, this threshold was usually set to 0 providing a binary output

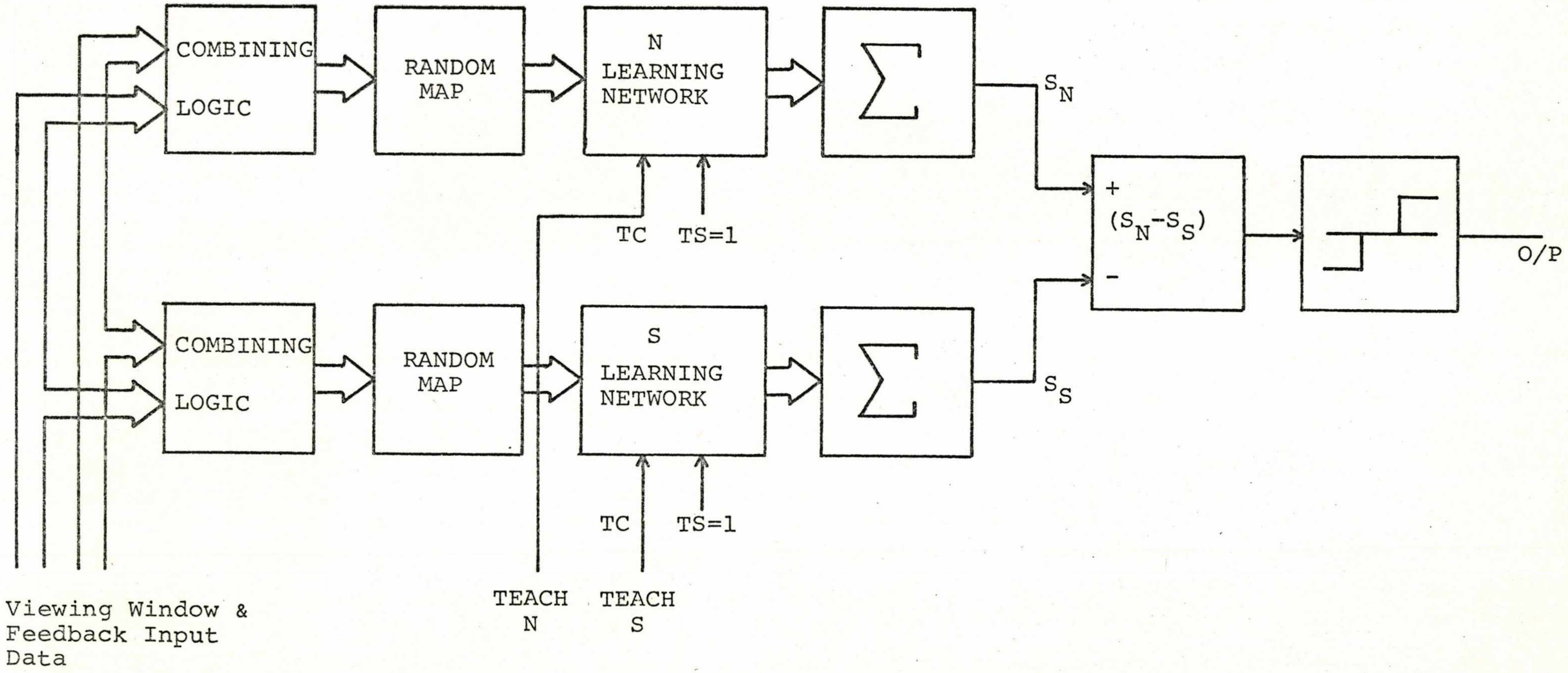


FIGURE 3.3.1 MAXIMUM RESPONSE LEARNING MODULE

(see section 3.2.1).

The input, which consists of the pattern from the viewing window combined with the feedback if there is any, is connected to the learning network via a random map so that the SLAMs sample random 4-tuples from the input space. This is considered in more detail in section 3.3.5.

3.3.2 Performance criteria.

In evaluating the performance of the different tracking methods a set of criteria have been established. These criteria are as follows:

1. Generalisation. This is the ability of the system to track unseen patterns after teaching with the same tracking path that it has been taught. A good tracking system should be able to accept a wide variation in the dimensions of the input patterns i.e., the more it generalises the better.

2. Tracking ability. This is an evaluation of the complexity of the patterns which the tracking system can be taught to track.

3. Amount of teaching required. This can usually be expressed by the number of cycles of tracking path of a pattern which must be tracked reliably. Sometimes, however, small corrections need to be made while the system

is running and these have also been noted when relevant.

4. Ease of teaching. This is a subjective evaluation by the teacher on how easy it is to direct the system around the desired tracking path. It must be stressed that the teacher interacts heavily with the system and the performance of the system is very dependent on the ability of the teacher.

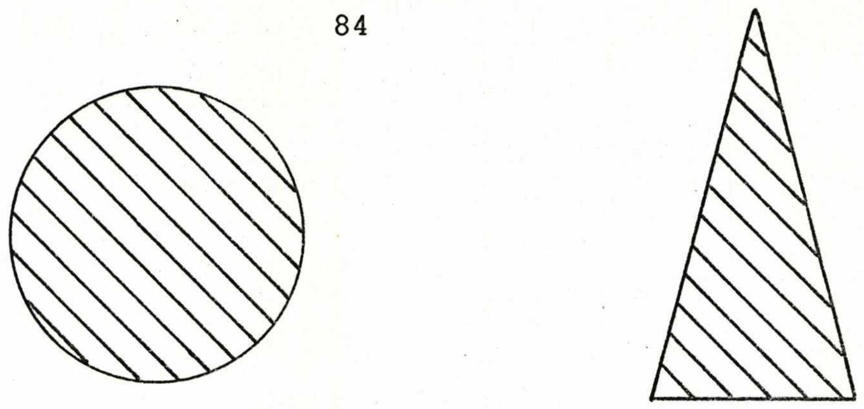
3.3.3 Performance of the system without feedback.

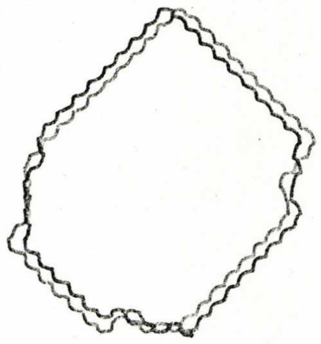
Without feedback the system should be able to be taught to follow edges. This was attempted with the smallest size of learning network available which was 64 SLAM-16s per learning network (i.e., a total active store of 4,096 bits). Slightly more stable teaching and tracking was observed when twice this number of elements was used. The performance of this system for the edge following task is given below.

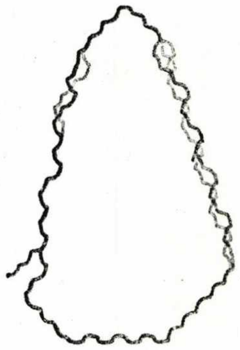
1. Generalisation.

Once taught a pattern, the system would track almost any other pattern hence, it has excellent generalising properties.

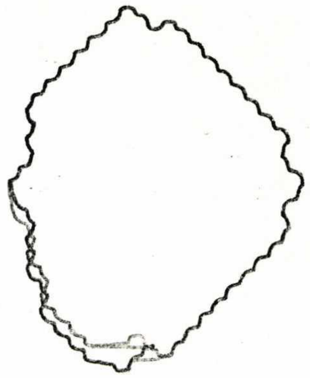
2. Although it could perform the edge following task very well, this was not considered a difficult task. The limitations of this system are in the way it tracks. This is illustrated in Fig. 3.3.2. If the system is first taught to track the circle and then is tested with the untaught triangle, then the tracking path cuts across

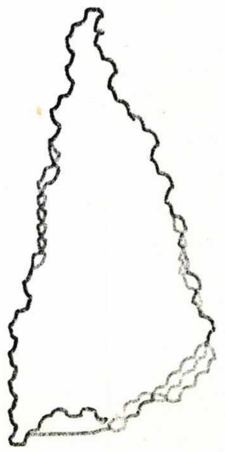






Tracking paths after training with O





Tracking paths after training with Δ

FIGURE 3.3.2.

the acute angle of the triangle. If only the triangle is taught and it is tested with the circle, then the tracking path becomes elongated at the top due to the teaching of the acute angle of the triangle. If the system is taught both patterns then it will track both properly.

3. Amount of teaching.

If taught one cycle around a pattern well then the teaching is sufficient. Poor teaching may take several cycles.

4. Ease of teaching.

The system was very easy to teach and any errors made were easy to correct.

Hence, this tracking system without feedback is very good in all respects but it is only suitable for edge following tasks.

3.3.4 Performance of the maximum response system with last step feedback.

The learning module receives information in the form of a 16x16 matrix which consists of the viewing window information combined with the feedback information. Three different methods of combining the feedback information with the viewing window information have been investigated.
A parameter,the feedback %, is preset for an experiment and

defines the amount of the input matrix over which the feedback function is effective. The actual bits of the input matrix which are to be associated with the feedback are randomly selected when the 'feedback %' is defined. The three feedback functions which have been investigated are as follows:

1. Replacement connections.

This function replaces the selected bits of the input matrix with feedback information.

2. Exclusive OR connections.

This function replaces the selected bits of the input matrix by the value of these bits exclusive ORed with the feedback information.

3. AND - OR connections.

This function replaces the selected bits of the input matrix by the value of those bits ANDed or ORed with the feedback information. Both AND and OR gates are used with this method because of the nature of the feedback information. For example, if with last step feedback the last step was N then all the N/S feedback connections will 1 and if the last step was S they would be zero; hence, if only AND gates were used the input matrix would be unaffected when the last step was N and all the N/S selected bits would be zero when the last step was S. To preserve a balance, when this function is selected, exactly half of the feedback functions are AND and half are OR.

These three functions have been tested with different values for 'feedback %'. A standard test has been used to achieve this. The system is firstly taught to track a square and it is then tested with a circle and a triangle. This is illustrated in Fig. 3.3.3 where the tracking paths obtained with this test are shown. The test was conducted with 20% replacement connections feedback with 128 SLAM-16s per learning network. Fig. 3.3.3(a) shows the tracking path obtained when teaching the system the square and (b) shows the path obtained when testing the system with the same square. The tracking path obtained when presented with the circle and triangle without further teaching are shown in (c) and (d).

Although tracking was possible with the smallest available size of learning network, i.e., 64 SLAMs per learning network, better results were obtained when twice this size was used and this was the size used to obtain the performance given below (i.e., 128 SLAM-16s per learning network which involves a total active store of 8,192 bits).

1. Generalisation.

If a value of feedback is within ±10% of the optimum value for a particular type of feedback then the generalisation with respect to the circle and triangle is good. In most cases they are both tracked successfully and in the others there are usually one or two critical points where the same errors are consistently made. (These may easily be corrected with a little extra teaching).

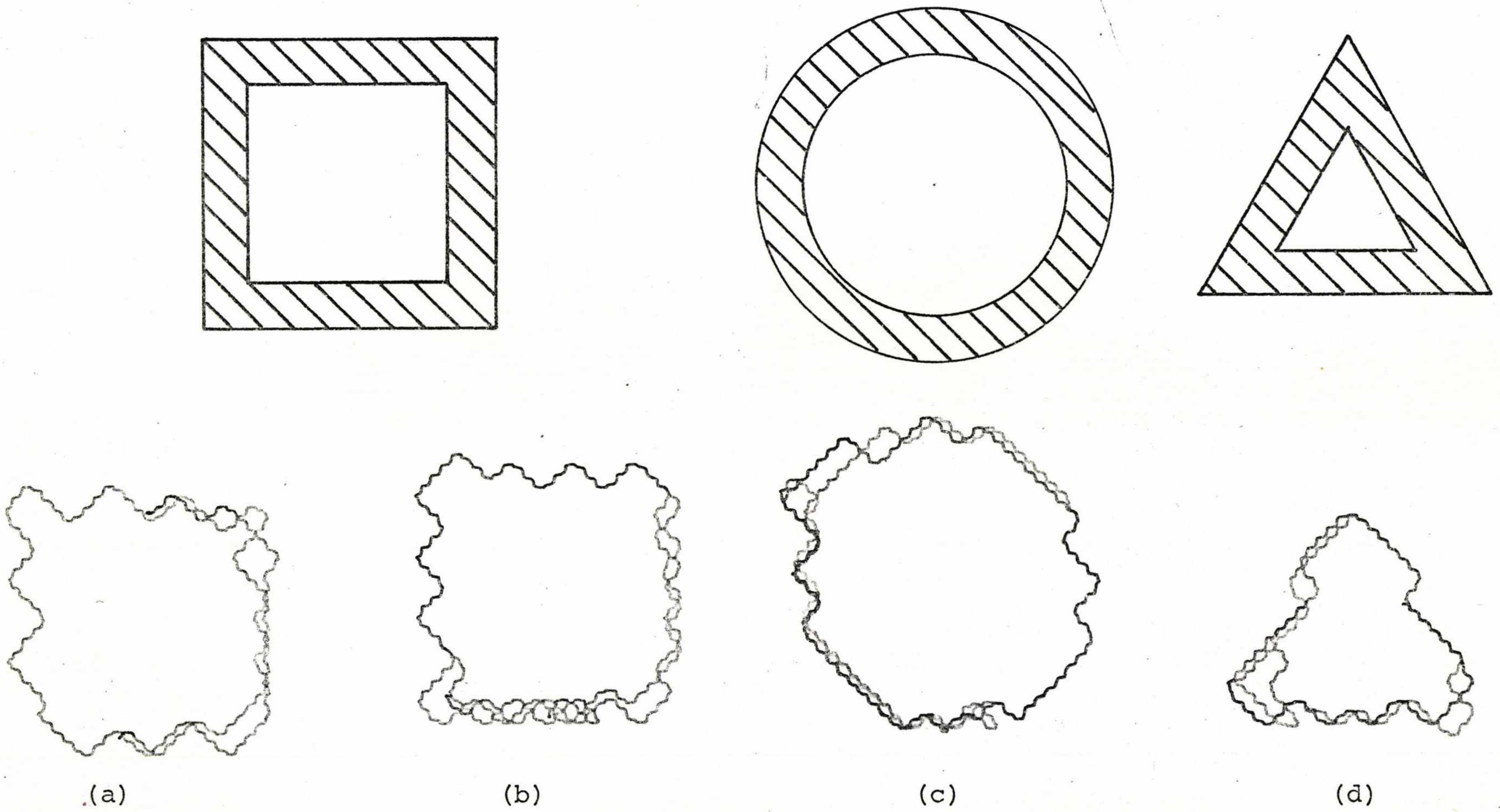


FIGURE 3.3.3.

2. Tracking Ability.

With the optimum value of feedback the system could be taught to track the patterns in Fig. 3.2.5. The limits of the 'feedback %' which could be used to track a square were found and these are as follows:

	Feedback %	
	Minimum	Maximum
1. replacement connections	3	75
2. Exclusive-OR connections	5	100
3. AND - OR connections	5	100

3. In most cases, except near the extreme limits indicated above, the square was taught for one complete cycle and this was sufficient. When any correcting was needed on the second cycle this was usually due to the teacher badly teaching during first cycle.

4. Ease of Teaching.

Near the mid values of the usable feedback % the teaching was very easy and it became increasingly more difficult as the limits of 'feedback %' were approached.

Using this system with reasonable values of feedback the patterns in Fig. 3.2.5 can be tracked. The three methods of applying feedback all worked well and no difference in performance could be detected between them.

In all the experiments mentioned in this chapter with last step feedback, 20% feedback with replacement connections were used unless otherwise stated.

3.3.5 The effect of different random maps.

For the above tests a different random map was used for each of the four learning networks. Some of these tests were repeated with the same random map for all four learning networks. This has the advantage that only one mapping has to be done instead of four. The mapping process takes a lot of time and by using one map instead of four the time taken to cycle a square was reduced from 25 to 9 seconds.

The structure of the learning module with one map is shown in Fig. 3.3.4. In this form each SLAM in the N learning network has a corresponding SLAM in the S learning network which samples the same bits of the input space.

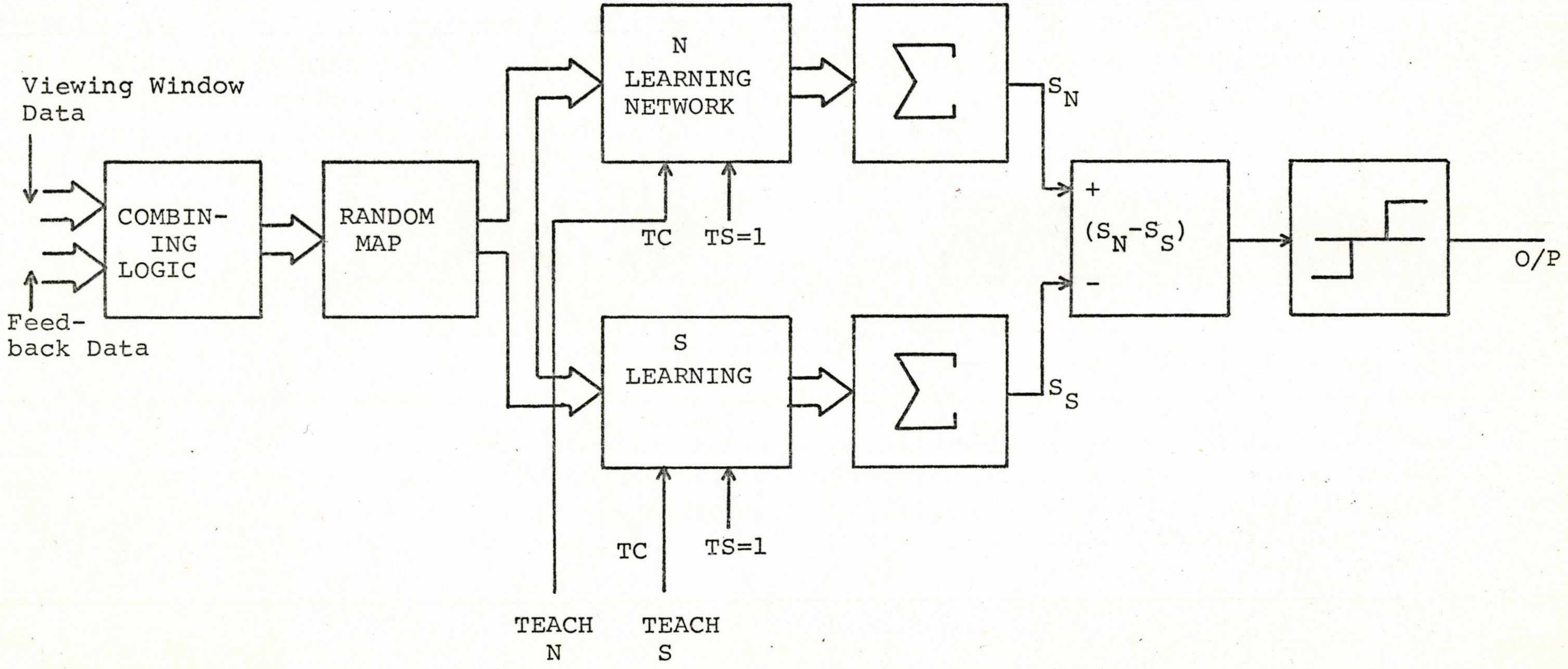

Viewing Window Data
Feed-back Data
COMBIN-ING LOGIC
RANDOM MAP
N LEARNING NETWORK
S LEARNING
TC
TS=1
TC
TS=1
TEACH N
TEACH S
Σ
Σ
S_N
S_S
+
(S_N-S_S)
-
O/P

FIGURE 3.3.4

This is similar to the structure of the T.R. SLAMs described in section 5.5. In fact, where a two-level threshold is used at the output, the performance is identical to a single T.R. SLAM learning network with a 2-level threshold.

Using a single map for the input data no significant change in the performance was detected.

One further modification has been made to the mapping to ensure that for less than 25% feedback each SLAM has either 1 or 0 feedback inputs. For if more than one feedback input of the same type is fed to the inputs of a SLAM this is equivalent to connecting these two inputs together and a SLAM-16 would effectively behave as a SLAM-8. When the random map was structured in this way no significant change in performance was detected and this single, structured type of map was used for all further experiments.

3.3.6 The performance of the system with damped feedback.

The damped feedback is generated by

$$NSV = NSV' \cdot (1-K) + K \cdot LD \ , \ (0<K\leqslant 1)$$

which is defined in section 3.2.3.

A value for K has to be preselected and for most tests this was set at 0.2. Which means that decisions made in the region 5 to 10 steps in the past are still considered. This type of feedback was investigated with twice the minimum size of learning networks (i.e., 128 SLAMs per module, a total active store of 8,192 bits) and 35% 'replacement connections' feedback. The performance of this system was as follows.

1. <u>Generalisation</u>

This system was used to track more complex patterns (e.g., with more than one loop) than had previously been possible. Once the system had been taught to track one of these patterns it could, in general, only accept very small changes in the shape or size of the pattern before tracking errors are made. The most common error is that it does not make the correct decision at a junction and the cycle which it does track does not cover all of the pattern.

2. <u>Tracking Ability</u>

Using this system it was possible to track all the patterns in Fig. 3.3.5 with the tracking paths shown. One pattern with three loops was taught but only with difficulty and the limit of this system appears to be at this level of complexity.

3. <u>Amount of Teaching</u>

Usually, if the system is taught for two to three cycles of the input pattern this is sufficient for that pattern to be tracked.

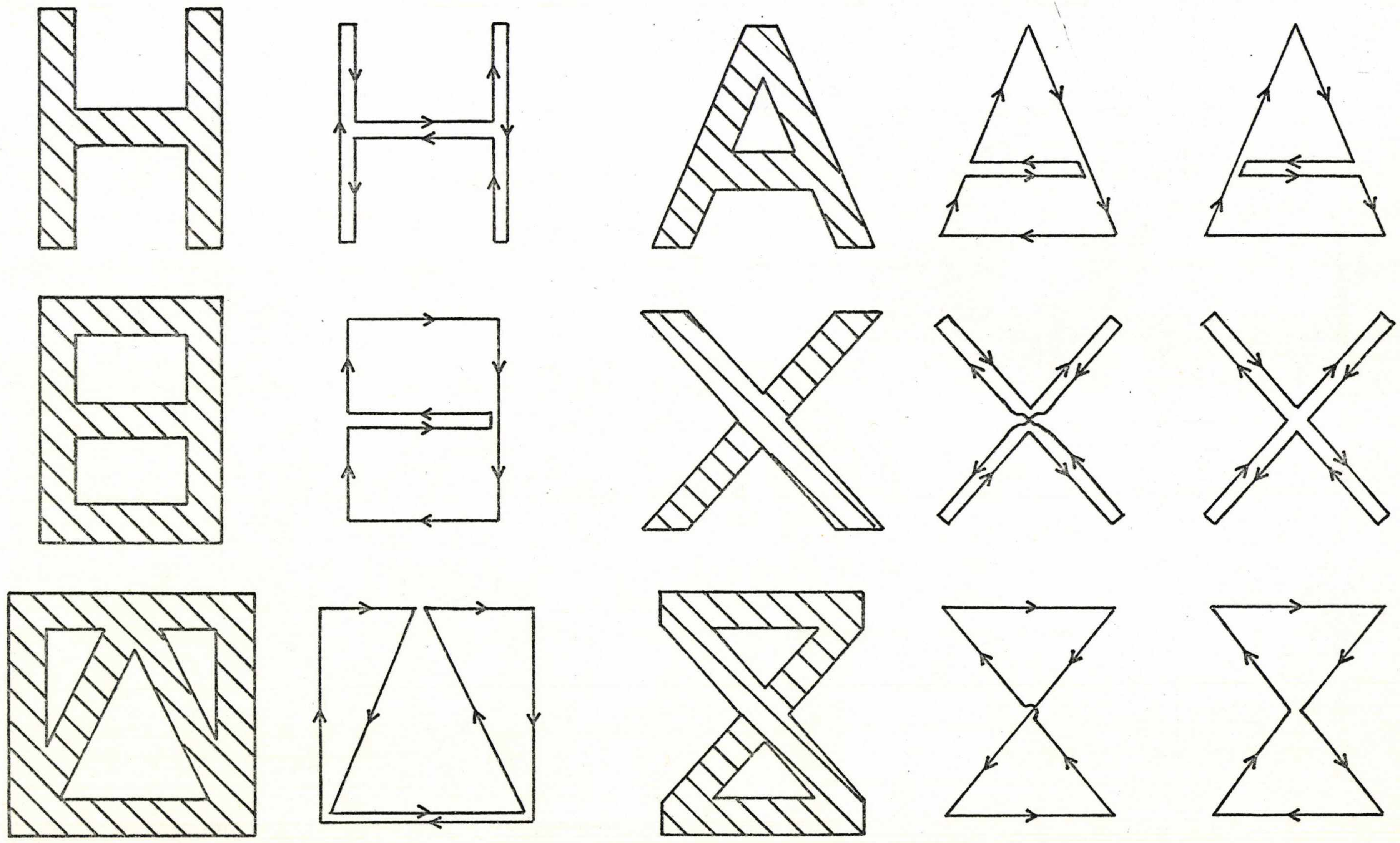


FIGURE 3.3.5

4. <u>Ease of Teaching</u>

In general, this system is reasonably easy to teach though not quite as easy as with last step feedback.

This system is able to track more complex patterns than was previously possible. To do this the 'feedback %' has had to be increased and the ability of the system to generalise has been greatly reduced.

3.3.7 <u>Saturation within the learning networks</u>.

With the maximum response structure with two learning networks per learning module it is possible that these learning networks may saturate i.e., both give 100% output for a particular input pattern.

A mechanism has been built into the system which will enable teaching to continue if saturation occurs. This mechanism teaches a randomly chosen 1/16 of the opposing learning network to output 0 for this particular input pattern. Hence, the correct response is now obtained. In practice, this mechanism was very rarely used and saturation usually only occurred for one of the two following reasons.

1. If the teacher is not very good and makes errors in teaching or if he changes his mind about the direction of the tracking path, then it is quite likely that he will contradict his own teaching and the learning networks will saturate.

2. If the tracking task is on the limit of the ability of the tracking system e.g., if the system is being tested near the limits of its feedback or if the input pattern is very complex, then it is quite possible that the limitation of the system will be indicated by the learning networks saturating. The saturation may be caused by a combination of the above two reasons and further careful teaching may achieve the desired tracking.

3.4 A PROBABILISTIC LEARNING SYSTEM.

3.4.1 Structure of the probabilistic learning module.

The probabilistic learning system uses the probabilistic learning network which is described in section 5.3. The probabilistic learning network is designed so that it can be taught to output both high and low responses.

The structure of the probabilistic learning module is shown in Fig. 3.4.1. With this system, only one learning network is required for each module.

With the probabilistic learning network a fraction of the SLAMs are taught with each clock cycle, this fraction is preset by the user. If less than ½ of the learning network is taught then one cannot be sure that the correct output response will occur after one clock cycle, (in general, 1/16th of the learning network is

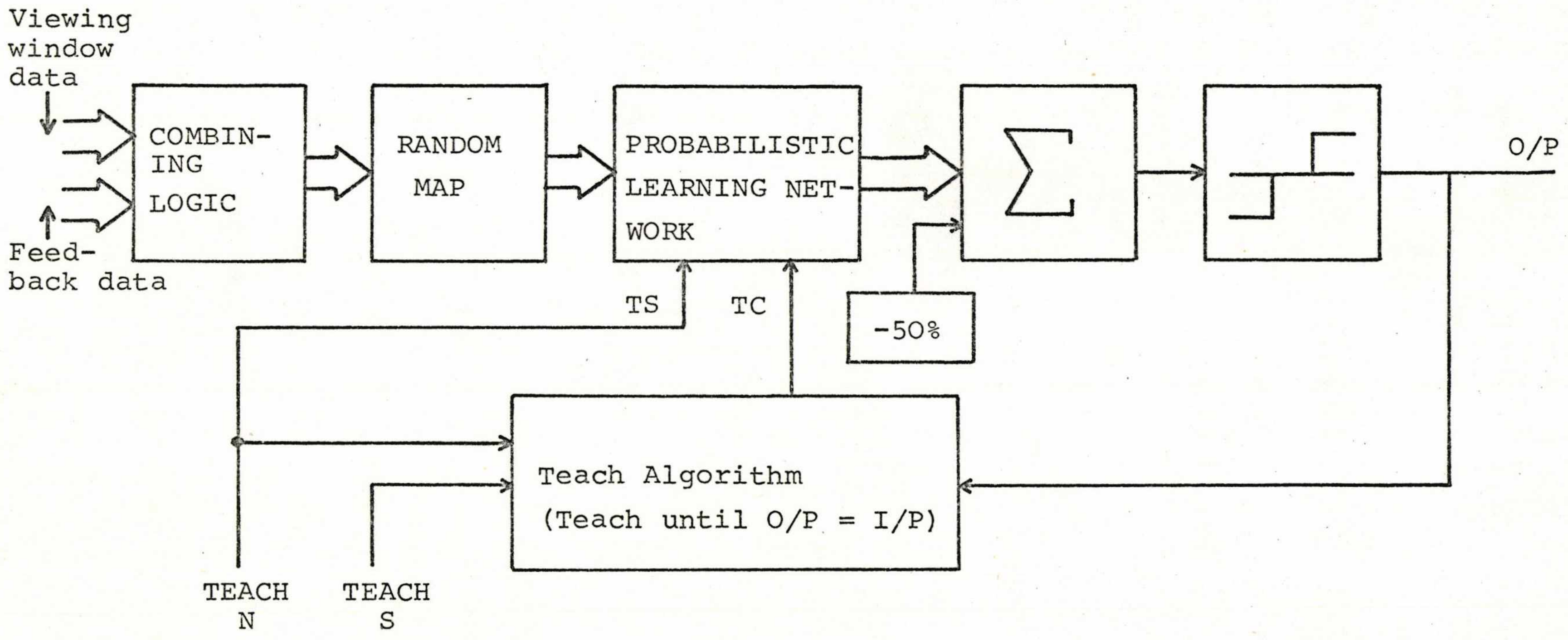


FIGURE 3.4.1 PROBABILISTIC LEARNING MODULE

usually taught per cycle). It is necessary for the learning network to output the desired responses when being taught, hence, a teaching mechanism is needed to ensure that the learning network is clocked a sufficient number of times for the desired response to be obtained.

To achieve this mechanism, two algorithms have been investigated which compare the output response of the learning network with the desired (non-zero) response and then clock the learning network until these responses are the same.

The first algorithm teaches the learning network once on receiving a teach command and then, if necessary, continues to clock the learning network until the desired response is achieved. This will be referred to as the 'teach first' method.

The second algorithm, on receiving a teach command, checks if the response from the learning network is correct and then, if necessary, clocks the learning network until the desired response is achieved. This will be referred to as the 'check first' method.

3.4.2 Performance of the probabilistic learning system without feedback.

With the smallest size of learning network (64 SLAM-16s per module) useful results could be obtained from the system and this is the size that was investigated in detail (i.e., 64 SLAMs per module which is a total active store of 2,048 bits). The following performance was obtained.

1. Generalisation

After teaching one pattern with either of the teaching algorithms, the system could track many other shapes hence, the generalisation is very good.

2. Tracking Ability

The tracking ability was similar to the ability of the maximum response learning system described in section 3.3.3. However, the probabilistic tracking system was more likely to make errors and drift-off or into the pattern.

3. Amount of Teaching

In general, it was necessary to teach two to three cycles of the pattern for it to be tracked.

4. Ease of Teaching

The system was easy to teach, though more difficult than the maximum response system.

Hence, the probabilistic learning system without feedback performs the edge following task well but is not, in most respects, as good as the maximum response method.

3.4.3 Performance of the probabilistic learning system with feedback.

Although useful results could be obtained with the smallest size of learning network, more consistent results were obtained if twice this size was used and this is the size which was investigated (i.e., 128 SLAMs per module which is a total active store of 4,096 bits).

The 2-level feedback was used with 'replacement' feedback connections and several different amounts of feedback were investigated. The performance with respect to line drawings was as follows:

1. Generalisation

In most cases the system was only just able to track a square after being taught it. When tested with a circle and triangle the only correct tracking occurred at the optimum values of feedback (20-30%). However, all the tracking of circles and triangles was very poor.

2. Tracking Ability

The tracking ability of this system is very limited. It was possible to teach the system to track around a square (using values of feedback between 10 and 70%). However, even this tracking was difficult to teach and more complex patterns were not considered possible. The 'teach first' algorithm would not work with this system and all the results were obtained by teaching with the 'check first' algorithm.

3. Amount of Teaching

A lot of teaching was required for this system at least 3 to 4 cycles of the input pattern.

4. Ease of Teaching

This system was the most difficult system investigated to teach. Lengthy careful teaching was required if a square was to be tracked.

Hence, this system is not very good for the line following task. It uses less active store then any other method but teaching the system is very difficult and the resulting performance is poor and uncertain.

3.5 A CUMULATIVE LEARNING SYSTEM.

3.5.1 Structure of the CL learning module.

The CL learning module uses the CL learning network which is described in section 5.4. Like the probabilistic learning network, the CL learning network is designed so that it can be taught to output both high and low responses.

The structure of the CL learning module is similar to the structure of the probabilistic learning module shown in Fig. 3.4.1 except that a CL learning network replaces the probabilistic one.

When a CL learning network is clocked the maximum change in output response is 1/16 of the total response. Hence, as for the probabilistic learning system, a teaching algorithm is required. The algorithms which were used for the probabilistic system, which are described in section 3.4.1, have also been investigated with the CL learning system.

3.5.2 Performance of the CL learning system without feedback.

The smallest size of CL SLAM learning network was sufficient for the edge following task. This involves 64 CL SLAM-16s per module i.e., a total active store of 8,192 bits. The performance of this system was as follows:

1. Generalisation

Both teaching algorithms were investigated and tracking could be achieved with either. However, if the 'check first' algorithm was used,although the system would reliably track the training pattern when tested with other shapes, errors were often made. If the 'teach first' algorithm was used then the generalisation was very good and many other shapes could be tracked.

2. Tracking Ability

Using the 'teach first' algorithm, the tracking ability was very similar to the ability of the maximum response learning system described in section 3.3.3.

3. Amount of Teaching

Tracking could usually be achieved after teaching two to three cycles of the pattern but this would be improved by further teaching.

4. Ease of Teaching

This system was very easy to teach and any errors made easy to correct.

Hence, the CL learning system without feedback can perform the edge following task well. The performance is comparable with the maximum distance system but twice as much active store is required.

3.5.3 Performance of the CL SLAM learning system with feedback.

As with other systems using feedback, more consistent results were obtained by using twice the smallest size of the learning network. This involved 128 CL SLAM-16s per module i.e., a total active store of 16,386 bits.

As with the probabilistic learning system, last step feedback was used with 'replacement' feedback connections and different amounts of feedback. However, for this system the 'check first' teaching algorithm did not work and all the results were obtained using the 'teach first' algorithm. The performance of this system with respect to line drawings was as follows:

1. <u>Generalisation</u>

The generalisation for other shapes was not very good for this system. When taught the square with a near optimum amount of feedback (20-30%) then in most cases the system would also track a circle and triangle. However, the tracking for the CL SLAM system was far better than the probability learning system.

2. <u>Tracking Ability</u>

The system could be taught to track a square with between 10 and 70% feedback and the performance of the system was similar to that of the equivalent maximum response learning system.

3. <u>Amount of Teaching</u>

This system required 3 to 4 cycles of the input pattern to teach it.

4. <u>Ease of Teaching</u>

This system was fairly easy to teach. However, it took more care and time to teach than the equivalent maximum distance system.

This system when using the 'teach first' algorithm has a performance which is comparable,though in general inferior,to the equivalent maximum distance system and it requires twice as much active store.

3.6 OBSERVATIONS ON THE RESULTS.

3.6.1 General performance of the system.

With no feedback all the learning modules investigated could perform the edge following task well using learning networks which covered the input matrix once only.

When last step feedback was added, so that lines could be tracked, it was found that in all cases better results were obtained by doubling the size of the learning networks. (Due to the addition of the feedback connections, this is the smallest size in which the whole of the input space is covered.) In both of the above cases, further increase in the size of the learning networks had very little effect.

The method of applying the feedback connections to the learning network was not critical. Also, the random map applied to the input was not critical.

With edge following, the generalisation was very good. However, when feedback was added and the input pattern was made more complex, this generalisation was reduced.

When damped feedback was used with patterns of several loops, the system was only able to track the training pattern reliably.

3.6.2 Relative performance of the learning modules. (Considering line following tasks).

Of the three learning modules tested the best performance in tracking and ease of teaching was obtained from the maximum response module. The structure of this module could be represented in a similar way to the structure of the other two modules as shown in Fig. 3.4.1 if the learning network was replaced by a T.R. SLAM learning network, and the function of the teach algorithm in this case would be to guard against saturation as described in section 3.3.7.

The input data is characterised by a long sequence of similar patterns. To demonstrate this all the patterns input to the learning modules while being taught to track a square were recorded, and used to obtain the following results. These patterns were divided into two groups, those for which the N/S learning module was taught N and those for which it was taught S. The histogram Fig. 3.6.1(a) shows the distribution obtained by comparing each pair of patterns in the N group and Fig. 3.6.1(b) shows the distribution of comparing each pattern of the N group with each pattern of the S group. The broad distribution for the N group shows that it consists of many patterns which are not similar in Hamming distance. There is a lot of overlap between the two distributions which indicates that it is not possible to distinguish between the two groups with a Hamming distance comparison to any archetypes. Therefore, the nets are doing something in addition to the above

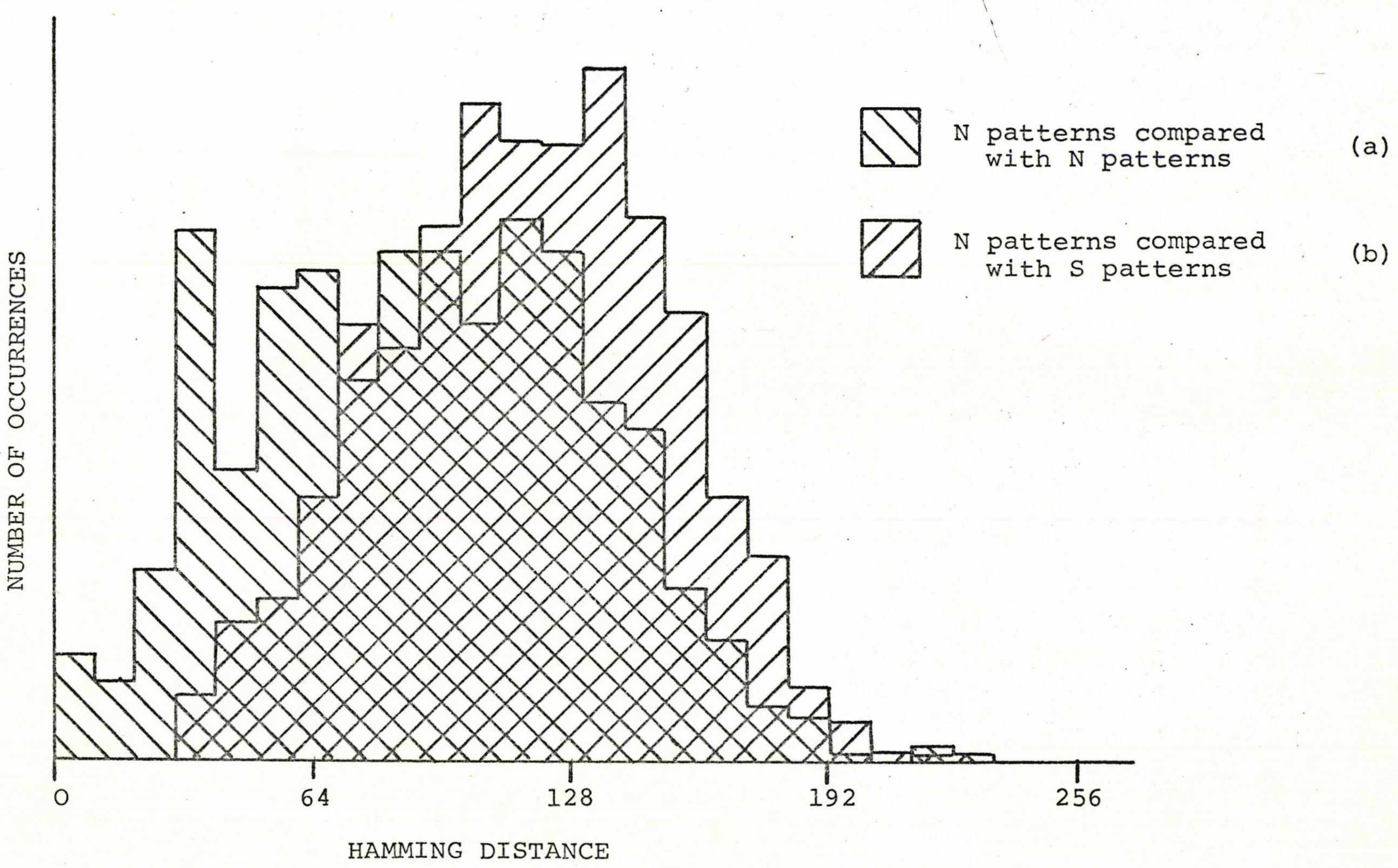

NUMBER OF OCCURRENCES
N patterns compared with N patterns (a)
N patterns compared with S patterns (b)
0
64
128
192
256
HAMMING DISTANCE

FIGURE 3.6.1

i.e., a few patterns close to those it recognises keep it "on track".

In Fig. 3.6.2 a particular decision during tracking is considered; (a) shows the pattern in the viewing window and the last direction moved. The distribution of this information to all the patterns in the N group is shown in (b) and for the S group in (c). Some of the patterns in (b) are very close in Hamming distance to the new pattern and this should be sufficient for the learning module to indicate that it belongs to the N group i.e., the next command from the N/S learning network should be N.

The better performance of the maximum response system is due to the large number of different input patterns taught for each class. The maximum response system does not forget any of the patterns it has been taught. However, both the probability and the cumulative learning networks have the property that consistently taught recent information may overwrite previously learnt information, and for tracking it is essential that all the patterns of a class be remembered with equal weight.

The amount of overwriting depends on the amount of overlap (i.e., where the Hamming distance is zero) between the classes for each learning element. If the order of the elements (i.e., number of inputs) was increased then this overlap would be decreased and one would expect that the performance of the probability and CL SLAM learning

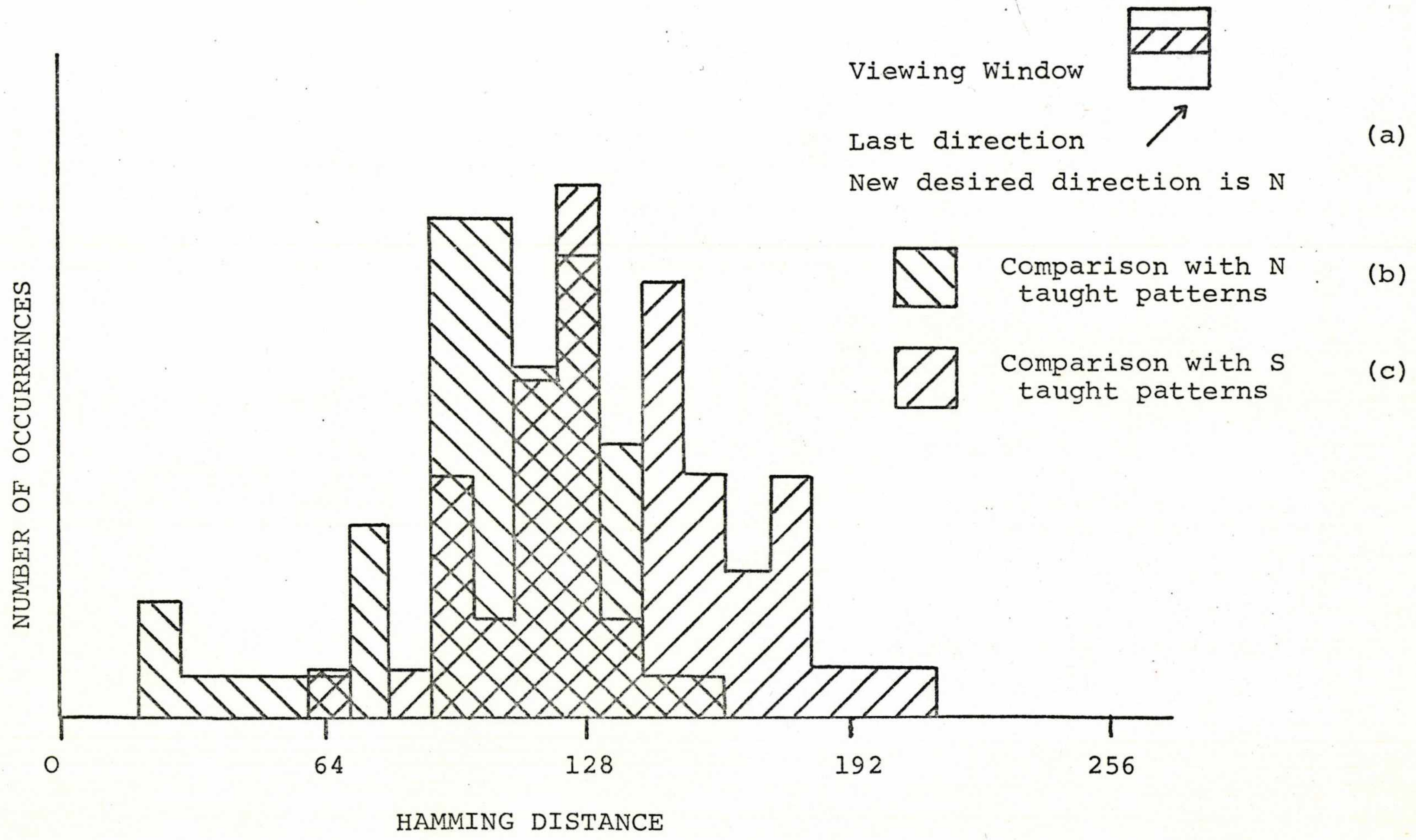

Viewing Window
Last direction
(a)
New desired direction is N
Comparison with N taught patterns
(b)
Comparison with S taught patterns
(c)
NUMBER OF OCCURRENCES
0
64
128
192
256
HAMMING DISTANCE

FIGURE 3.6.2

systems would improve with respect to the performance of the maximum response system.

The probability learning system would only work with the 'check first' teach algorithm. This is because all decisions of the learning networks were very critical (i.e., near the 50% mid value). If too many of one class was taught the learning network becomes permanently biased towards that class.

The CL learning system does not generalise well with the 'check first' tracking algorithm as the learning network needs to be taught, even when the response it outputs is correct, so that these responses may be increased and do not remain near the critical 50% value.

The CL learning system could be taught to track a square with a 'dead zone' of up to 25% in the threshold decision at the output of the learning network. The probability learning system could not be taught to track a square if there was a dead zone.

Using the maximum response learning modules, the system was taught to track a square and the number of elements which were taught to 1 at each step was recorded. These results have been plotted in Fig. 3.6.3 and they demonstrate that after the initial teaching of the square further teaching has very little effect.

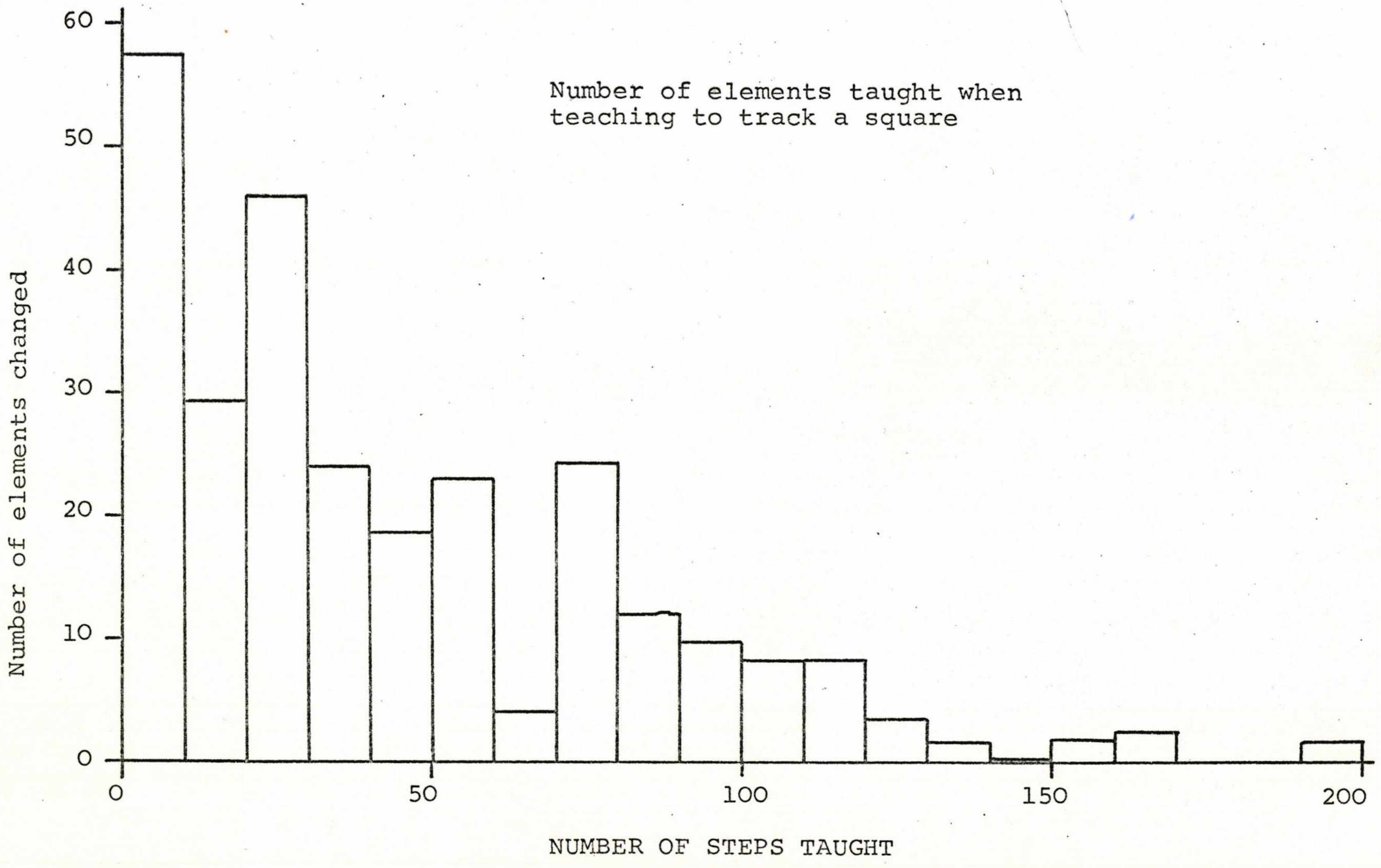

Number of elements taught when teaching to track a square
Number of elements changed
NUMBER OF STEPS TAUGHT
0
10
20
30
40
50
60
0
50
100
150
200

FIGURE 3.6.3

CHAPTER 4
THE CLASSIFYING SYSTEM

An attempt has been made to recognise patterns from the tracking information produced by the tracking system.

In the first technique which was investigated, the data from the tracking system was transferred into a shift register and the contents of the shift register were examined in parallel, with a learning network. This method was not very successful but is mentioned at the end of this chapter in section 4.4.

In the second method, the tracking information was fed serially into a learning network which, as a result of feedback, is sequence-sensitive, and this method is considered in detail in this chapter.

4.1 THE INPUT DATA.

4.1.1 Features of input data.

To create a structure for classifying patterns from tracking motions, it is important to understand how distinguishing features are contained in the tracking information.

The tracking information or 'input data' to the classifier has the following characteristics.

It is a sequence of events, the length of the sequence being dependent on the size of the pattern and the tracking path. (Each event consists of the information obtained from the tracking system in moving one step.) Typically, for a thick line drawing of a square, the sequence is the order of 100 events long for one cycle of the square.

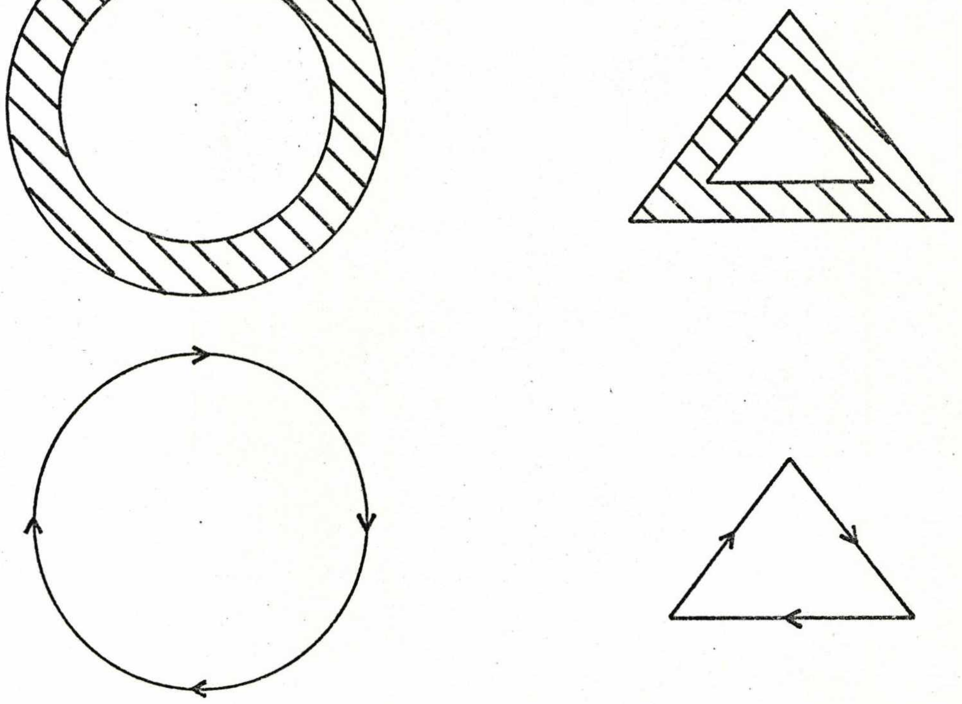
This sequence of events contains within it a set of features of the pattern such as straight lines, curves, corners, etc. each feature being several events long, and the features occur in a unique order in the sequence.

4.1.2 Test data.

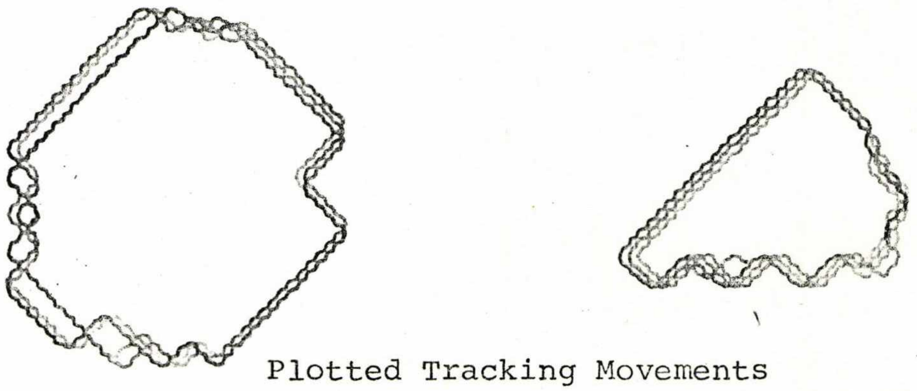
To assist the development and to evaluate the performance of the classifier, three different sets of test data have been used. This data is stored in the form of tracking movements on punched paper tape so that exactly the same data can be used for each test and the result of changing parameters can be observed. Each set of test data consists of the tracking information for two patterns. These are used as the training set to teach two classes to the classifier. This same data is also used to test the classifier. Each of the sets of data contains different characteristics and these are described below.

1. Test data (A)

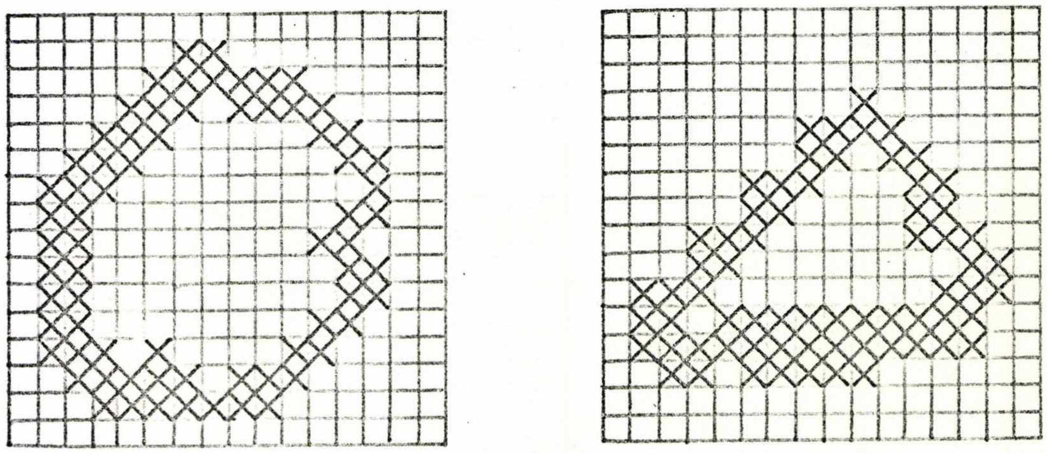
This is illustrated in Fig. 4.1.1 and is of a circle



Tracking Paths



Plotted Tracking Movements



Normalised tracking movements plotted on a 16x16 matrix

FIGURE 4.1.1 TEST DATA (A)

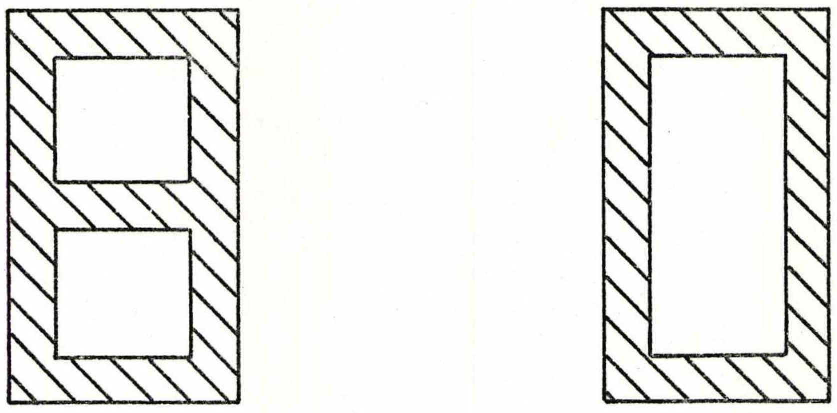
and a triangle. This data was obtained from the tracking system. One cycle of the circle requires about 90 steps and one cycle of the triangle requires about 60 steps. The main characteristic of this data is that the tracking paths are fundamentally the same form and the classifier must detect the small differences in the shape of the scan path to classify the patterns e.g., it must detect that the point of the triangle is different from the continuous curve of the circle.

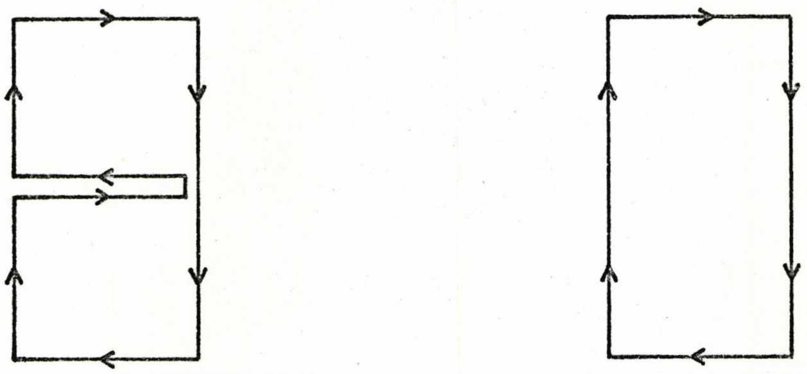
2. Test data (B)

This is illustrated in Fig. 4.1.2 and is of a rectangle with a center line and a rectangle without one. This data was also obtained from the tracking system. One cycle of the rectangle with the center line requires about 130 steps per cycle and the plain rectangle requires about 90 steps per cycle. The scan paths for the two patterns are the same except where the center line occurs in the first pattern. Hence, to classify these patterns, the classifier must detect the tracking motions relevant to the center line for the first pattern and must detect their absence at the equivalent position for the second pattern. All the tracking motions concerned with the rectangle alone should be ignored as they are common to both patterns. This is explained in greater detail in section 4.2.3.

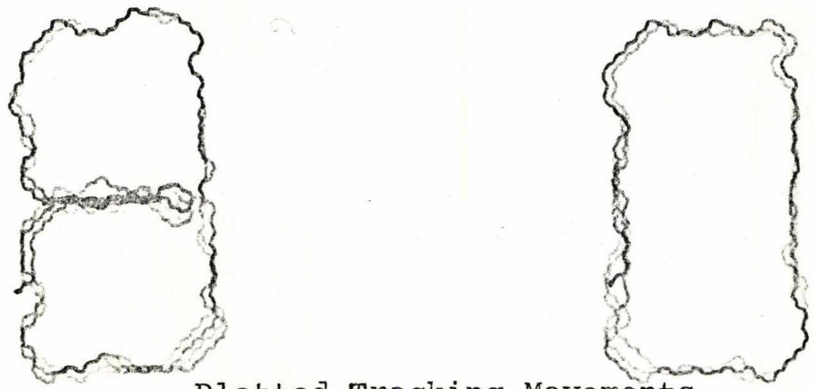
3. Test data (C)

This is illustrated in Fig. 4.1.3. for a square and a triangle. This data was calculated and generated by hand. The object of this data is to provide simplified versions

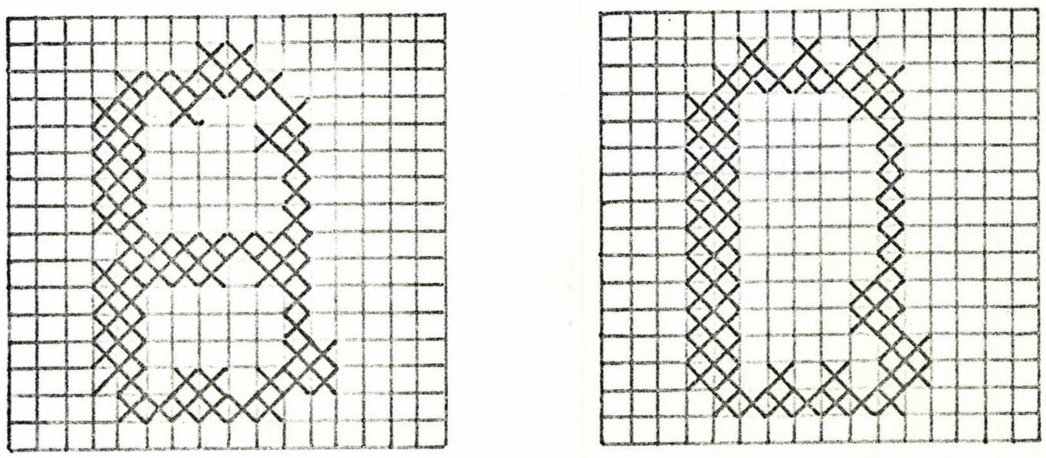




Tracking Paths

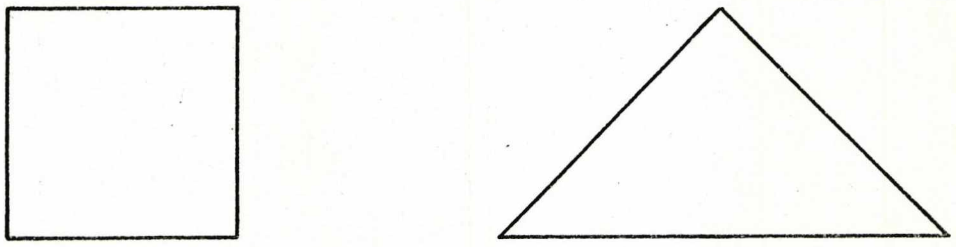


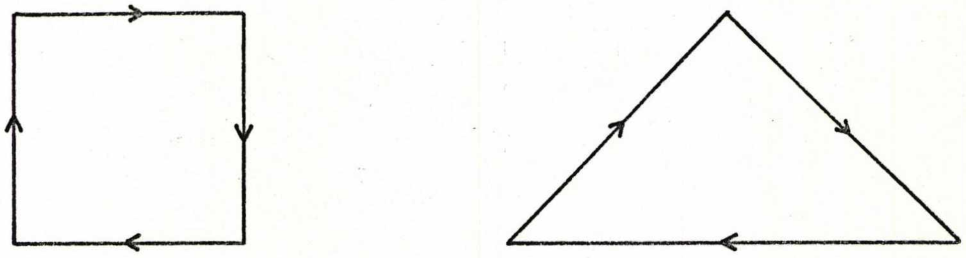
Plotted Tracking Movements



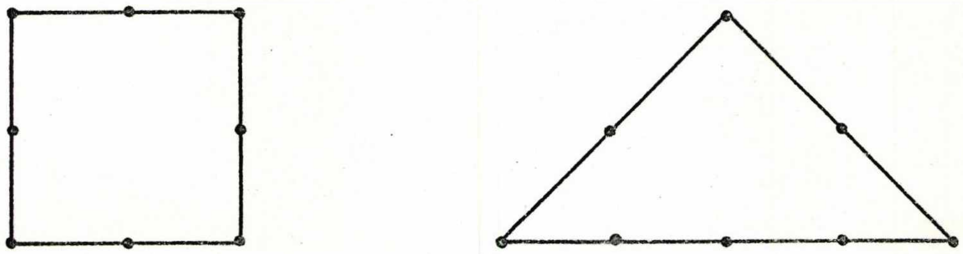
Normalised tracking movements plotted on a 16x16 matrix

FIGURE 4.1.2 TEST DATA (B)

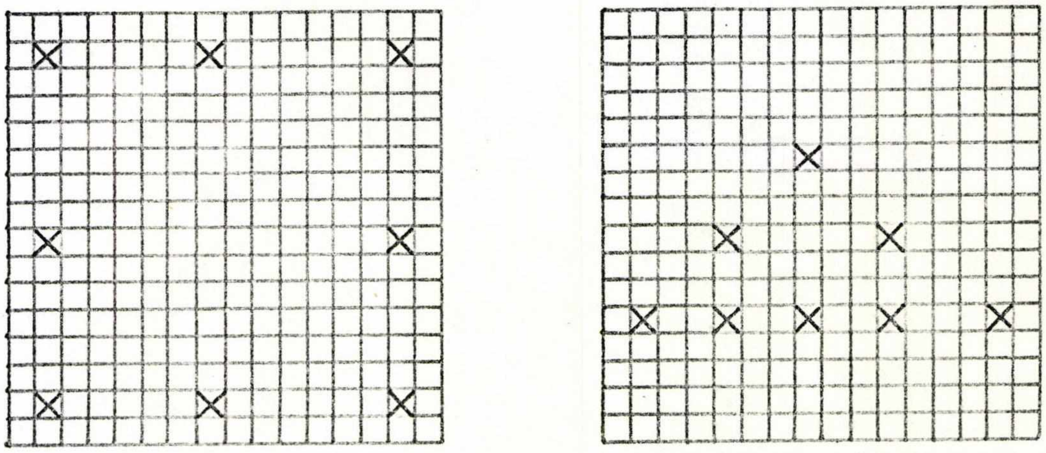




Tracking Paths



Plotted Tracking Movements



Normalised tracking movements plotted on a 16x16 matrix

FIGURE 4.1.3 TEST DATA (C)

of tracking motions with short, well defined cycles of 8 steps for each pattern. This data was used to assist the development of the classifier when it was not capable of producing useful results with test data (A) or test data (B).

4.1.3 Format of input data.

The tracking information is received from the tracking system in the form of a four-bit word for each step. Each bit, if set, represents a move in one of the four possible directions N. S. E. W. (If we assume that the viewing window must move at each instant of clocked time then it must move in one of 8 possible directions. Hence, 3 bits of information are obtained from the tracking system per step. However, the 4-bit code is a more convenient form to observe, and to input to a learning network.)

For the first experiments these 4-bit words were fed directly to the input of the classifier. Therefore, for a feature of a pattern, the classifier must have an internal memory for several steps. However, to obtain knowledge of the spatial position of the features it is necessary to detect in which order the features occur and for this a much longer internal memory is required. Several experiments were carried out with this type of input. The classifier did not work at all for the test data (A), however some results were obtained with test data

(C) in which both the shape of the features and the distance between features is only a few steps.

An important factor which must be considered with data taken directly from the tracking system is that it contains 'jitter' due to the characteristics of the tracking system. For example, if the system is tracking a horizontal line in the east direction, one might expect the sequence of commands to be E, E, E, E, ... The tracking system is, however, taught imperfectly by a human teacher to maintain its N/S position on that line so that a more likely sequence in practice would be NE, SE, NE, SE, ... Hence, a lot of N/S activity would be indicated which is not produced by the shape of the pattern.

In an attempt to overcome the problem of the long memory required and the 'jitter' problem mentioned above, the input information was averaged over several steps. The way this is achieved is illustrated in Fig. 4.1.4. The program written for this allows the average value of the last n steps to be evaluated (where $1 \leqslant n \leqslant 16$). For each direction the number of times that a move is indicated, V, for the last n steps, is evaluated; this number is then presented to the classifier as V bits set in a binary vector of length n. For each step of the tracking system, 4.n bits of information are presented to the classifier. The effect of this method is to generate an 'average' direction of tracking from the last n steps. This effectively eliminates the 'jitter'.

When n is large, some indication as to the spacial position of the features is now input to the classifier because the input vector cannot change very much between features. However, this also means that features are smoothed and in the case of the test data (A), for example, it is very important that features such as the points of the triangle should not be smoothed too much. This method was used with the classifier and although it still did not work properly with data from the tracking system, better results were obtained.

To avoid the smoothing problem a third method of inputting the data was developed. Instead of inputting the motions involved in tracking, it was decided to input the position of the viewing window with respect to the pattern. The advantage of this method is that features are spacially located by the input data, hence a long memory within the classifier is not necessary and the internal memory is only needed to detect the features. There is a practical difficulty in obtaining this data because the information from the tracking system is in the form of tracking motions. To generate position from these motions, a scaling system was used. This analyses the motions from the tracking system for the first cycle of the pattern (without transmission to the analyser) so that the limits of the pattern could be determined. Then, after each motion is input, the position is normalised with respect to these limits. This scaling enables the pattern to be defined on a standard matrix size. The positions as they are generated can be used to generate the tracking path that the tracking system follows. The standard size has been defined on a 16x16 matrix and the results of plotting these motions on this matrix are shown in Figs. 4.1.1, 4.1.2 and 4.1.3.

Hence the output from this scaling system is one bit in 256 i.e. 8 bits of information per step. This was coded into two 16-bit binary vectors in which the X and Y coordinates of position on the normalised matrix are indicated by the number of bits set. This is illustrated in Fig. 4.1.4. When using this data form for the input to the classifier, a further improvement in the performance was noted and this form has been used for all the development of the classifier. Although in the above description, the positions were generated by software, this is not necessary if one considers the tracking system and classifier combined. Then the addresses that are sent to position the viewing window of the camera could be used to generate the classifier input.

4.1.4 Summary.

In this section the nature of the input sequence has been discussed in detail. The input sequence to the classifier is a sequence of positions, each position being defined by two 16-bit vectors.

The features of the pattern are defined by several

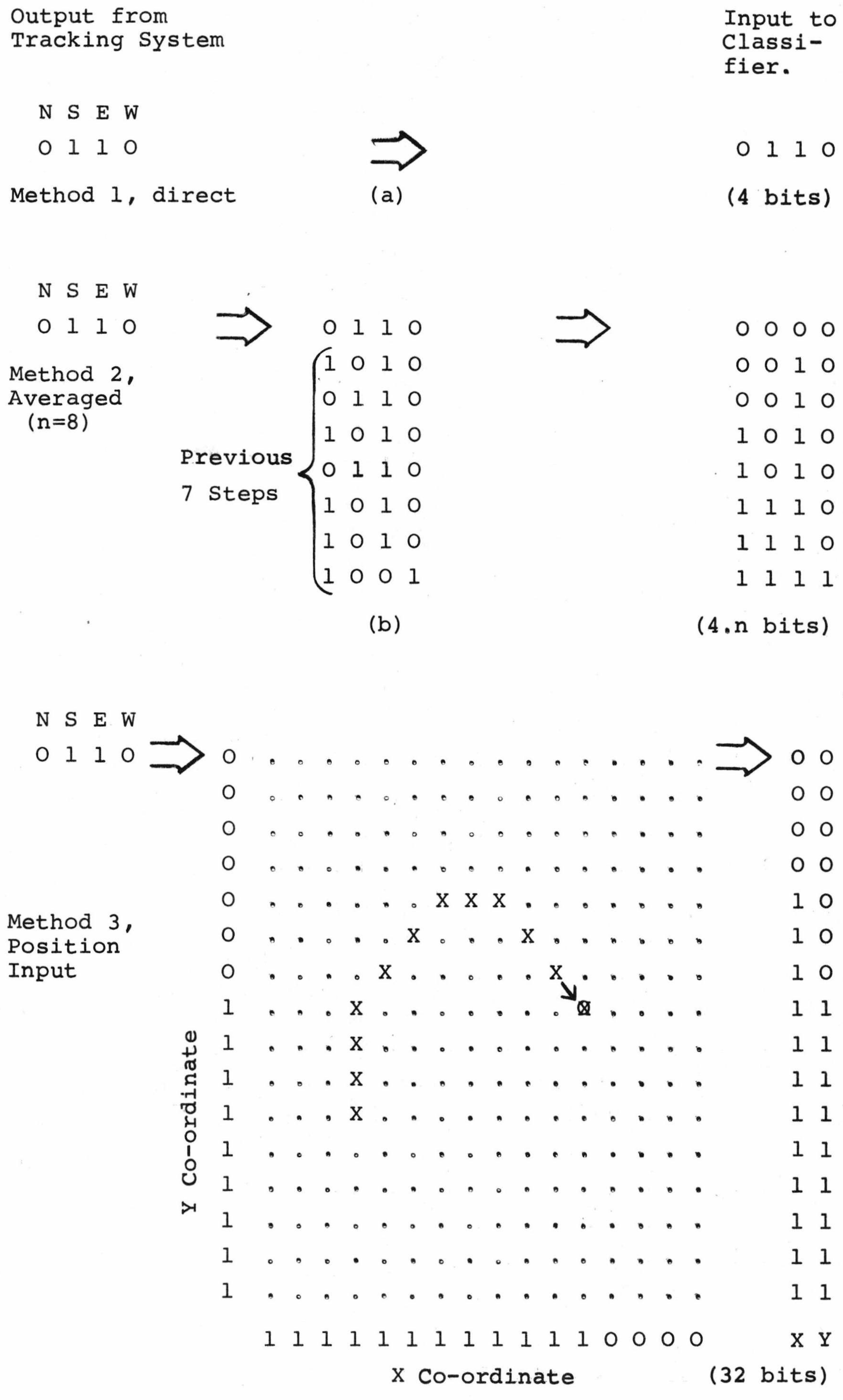

Output from Tracking System
Input to Classifier.
N S E W
0 1 1 0
Method 1, direct
(a)
0 1 1 0
(4 bits)
N S E W
0 1 1 0
Method 2, Averaged (n=8)
0 1 1 0
1 0 1 0
0 1 1 0
1 0 1 0
0 1 1 0
1 0 1 0
1 0 1 0
1 0 0 1
Previous 7 Steps
0 0 0 0
0 0 1 0
0 0 1 0
1 0 1 0
1 0 1 0
1 1 1 0
1 1 1 0
1 1 1 1
(b)
(4.n bits)
N S E W
0 1 1 0
Method 3, Position Input
Y Co-ordinate
0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
X Co-ordinate
X Y
0 0
0 0
0 0
0 0
1 0
1 0
1 0
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
(32 bits)
(c)

FIGURE 4.1.4

steps of the input sequence,hence the classifier must have some short-term memory to store them.

To test the classifier three sets of test data have been established. Set (A) for problems where two scan paths have only minor detail differences, see **Fig.** 4.1.1. Set (B) is typical of problems where one part of the scan path is different for one class where the rest of the scan paths are identical. Set (C) has only 8 steps per cycle and is used to evaluate systems for which the test data (A) and (B) is too long.

4.2 STRUCTURE DEVELOPMENT OF THE CLASSIFIER.

4.2.1 General system.

The general structure that was chosen for the classifier is shown in Fig. 4.2.1. The aim of this structure is to associate the output codeword with a class of input sequences, so that when an input is applied to it, after training, the appropriate codeword will form at the output.

The sequential data from the tracking system is randomly connected to some of the inputs of the classifier learning network and there is internal feedback around the network to the rest of the inputs.

During training, information from the input, and from a codeword which defines the class, is fed to the

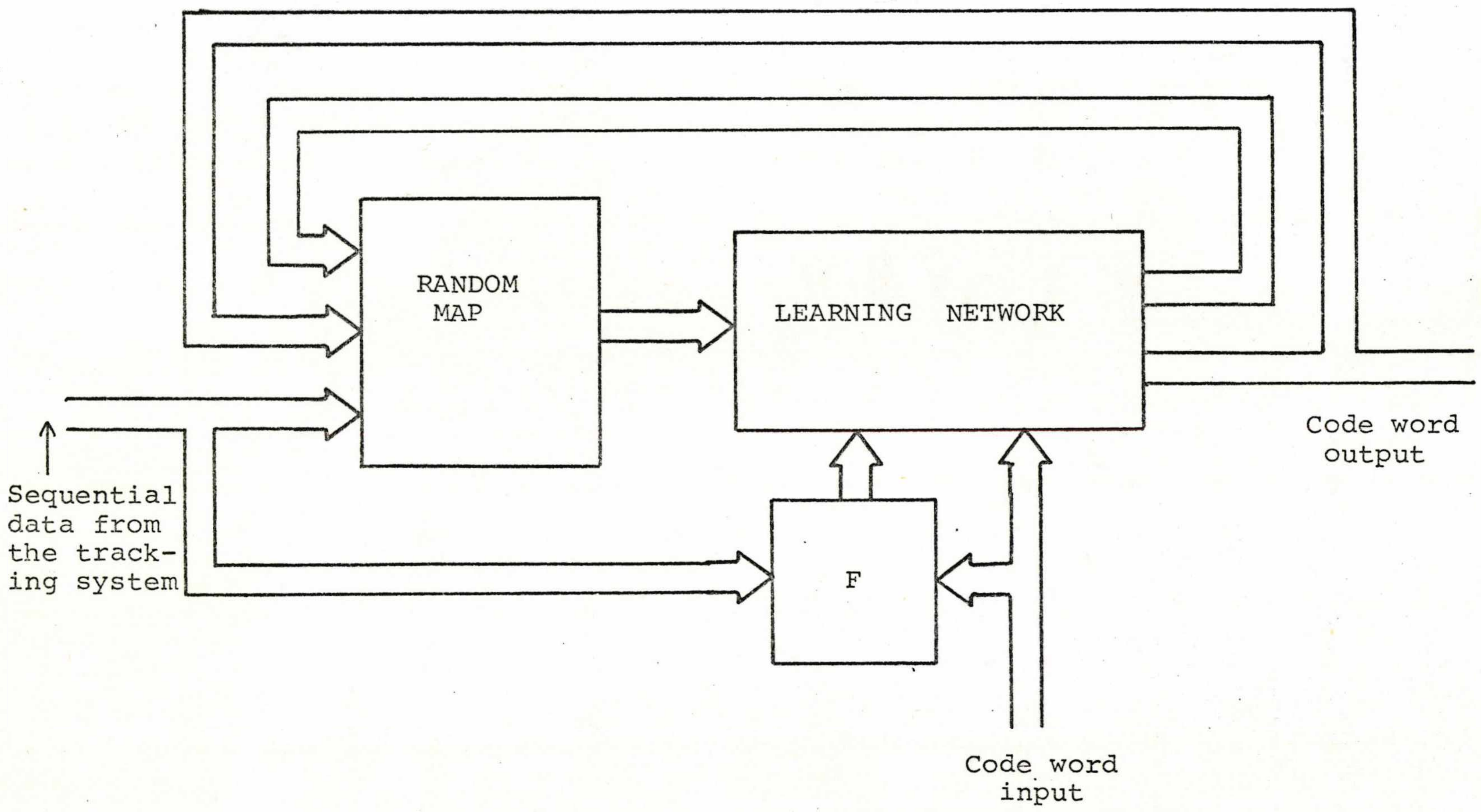

RANDOM MAP
LEARNING NETWORK
F
Sequential data from the tracking system
Code word output
Code word input

FIGURE 4.2.1

teach terminals and the SLAMs are clocked (i.e., taught). The SLAMs which have the codeword only applied to the teach terminals are the ones which should **regenerate** the codeword when the input sequence is applied.

The other SLAMs receive teach information from both the input data **and** the codeword combined together. The combining is achieved by combinational logic gates (shown by the box 'F' in the diagram). The function realised by this logic is very important and will be discussed later.

Three main measures are used in discussing the properties of the classifier and these are defined as follows:

Store penetration: This is the number of binary locations in the SLAM stores which are accessed (i.e. are set to 1 or 0) when the classifier is taught. (It is usually stated as a percentage of the total SLAM store size.)

After one cycle of an input pattern has been taught, one would expect very little increase in this penetration. This measure does not take into account the number of times that a bit is changed during teaching.

Store Overlap: This is the number of binary locations of the SLAM stores which, after being accessed by teaching a first input pattern, change their value on being taught a second input pattern. (It is usually stated as a percentage of the store penetration of the first pattern taught). This measure depends to some extent on the position in the cycle of each pattern that teaching is terminated.

<u>Pattern Activity</u>: This measure is not rigo~~u~~rously defined and is used in a qualitative sense only. It is used to compare two sequences of patterns and is based on both the bits which change during the sequence and the correlation between the bits which change value. Hence a sequence of patterns is said to be more active than another if either the number of bits which change state is higher or the correlation between bits which change state is lower.

4.2.2 Development into a two-net structure.

The first step in the development of the general structure was to realise that it contains two learning networks which perform different functions. The second network which is taught only the codeword is to detect states of the structure and decide which codeword to output. The first network, which is taught a combination of input and codeword information, is intended to contain the short-term memory necessary for detecting distinguishing features and generates the states which are detected by the second network.

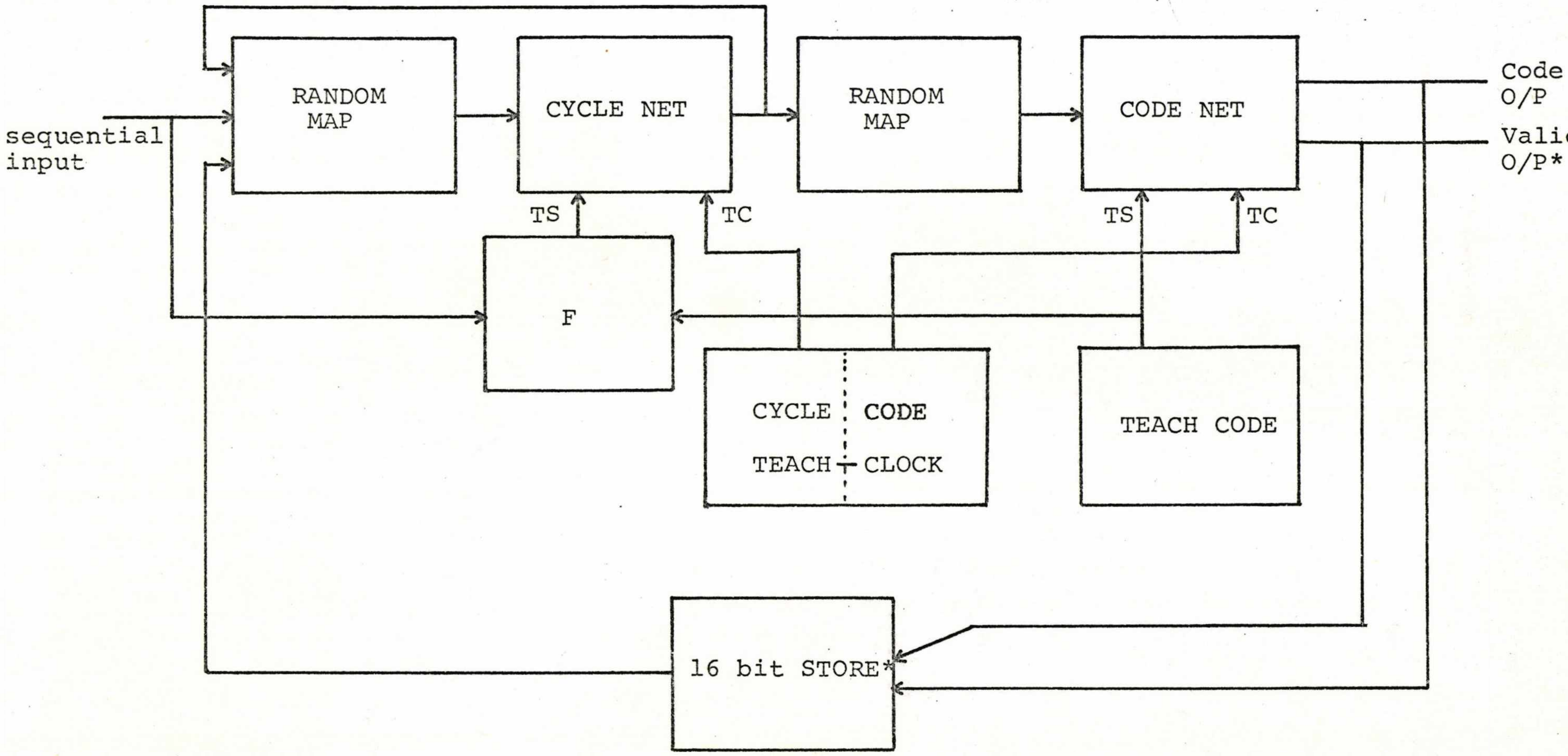
The structure of the classifier with the above development is shown in Fig. 4.2.2. The first learning network is called the cycle net and the second one is called the code net.

The cycle net has feedback around it so that it may have a memory of several steps to detect features (see section 4.2.4). There is a second feedback loop from the output of the code net to the input of the cycle net. This feedback has two main functions: it will be shown to reduce the amount of contradiction due to the teaching of several classes within the cycle net and it also should enable a codeword to become stable at the output. These properties of this feedback are discussed in more detail in section 4.3.4.

The teach clock for the structure has been split into two sections, this allows the two learning networks to be taught separately.

4.2.3 Development of the code net

It would appear more logical to discuss the development of the cycle net first, however, the exact function of the cycle net is not easy to define and is dealt with in the next section. The function of the code net is easier to define, it was developed first and will now be considered first. The general function of the cycle net is to enter state cycles in sympathy with the



* used with TR SLAM Code Net

FIGURE 4.2.2

input sequence. These cycles should pass through states related to the class of the input sequence.

The code net must detect when these unique states occur and output a codeword which will indicate the class of the input sequence.

The first structure of the code net was a learning network of 16 SLAM-16s. The clock terminals were connected together and each of the teach sense terminals was connected to one bit of a 16-bit codeword. During training all the SLAMs are taught for every step of the input sequence.

Now consider the nature of the data which is presented to the code net. This data is considered in the form of 4-tuples randomly sampled by the code net. When the system is tracking patterns of the test data (B) type, for example, then it is probable that except in the region of the center line where the features are different, the information at the output of the cycle net will be very similar for the two classes. Also, as the cycle net is to go into cycles, one would expect a lot of activity (as defined in section 4.2.1) at the outputs. The result of both the above factors is that one would expect that many 4-tuples of outputs would have common values at times for different classes.

Hence, the input to the code net is a large number of different patterns for each SLAM only a few of which define classes and many of which are common to both.

Using the structure for the code net described above, where every input pattern is taught, the net will always be heavily overtaught with respect to the last class taught as follows. In such cases, all the input 4-tuples which are common between classes are taught to output the code of the last class taught.

A feature of the code net is that it is possible to observe the output before teaching at each step, and evaluate how similar that output is to the desired codeword without teaching. An algorithm was written to utilise this fact to overcome the above overteaching problem. This algorithm finds the difference between the output of the code net and the desired codeword and then corrects a fraction of the difference, by teaching this fraction to the code net. This algorithm produced an overall improvement in the performance of the classifier. However, it does have several disadvantages as follows.

1. The algorithm is time (i.e., sequence length) dependent and if many cycles of a pattern, or many patterns of a class, are taught, then the overteaching will still occur since, with time, complete teaching is approached.

2. Only a portion of the codeword is taught and hence, only this form can be fed back which greatly reduces the effect of this feedback.

3. The above feature also implies an additional restriction on the number of distinct classes

taught. More classes simply lower the Hamming distance between codewords which might cause non-distinct code-words to be taught.

This algorithm was abandoned because of the above disadvantages.

There is a need for a different type of learning element for the task required of the code net. A new element the 'T.R. SLAM' was developed to do this, and it is described in detail in section 5.5.

This element has an extra output which indicates whether the normal output is valid or invalid (i.e., not taught or overtaught). So this element is insensitive to overteaching by the last class taught. The feedback from the output of the code net has been modified by a 16-bit memory so that only the last valid output is fed back. This has been added because for the majority of the time while the cycle net is establishing a definite cycle, the outputs from the code net are not valid. This is also discussed in more detail in section 5.5. Fig. 4.2.2 shows the structure of the classifier with T.R. SLAMs in the code net. When using T.R. SLAMs twice the active store required for normal SLAMs is used. For the code net of 16 T.R. SLAM-16s described above 512 bits of active storage are required.

4.2.4 Development of the cycle net.

It has been stated (section 4.2.3) that the general function of the cycle net is to enter cycles in sympathy with the input sequences and in so doing pass through unique states relevant to the class of the input sequence.

This is necessary because, for some classes, the classifier must remember several previous steps to be able to react to an important feature. This can be demonstrated by an example. Consider the patterns in the test data (B) and assume that there is no memory of past events in the classifier. When tracking the first pattern the centerline results in several unique positions with respect to the pattern. Hence, the classifier can come to a definite decision. However, in tracking the second pattern all the positions the tracking system passes through are common to both patterns. Hence, only arbitrary decisions can be made.

The only way to recognise the second pattern is to notice that the tracking motions do not go along a center line, i.e., the absence of a feature must be 'perceived', and the only way to do this is to provide the classifier with some memory of past events: hence, the feedback in the cycle net.

A similar type of argument can be applied to the test data (A) patterns. Here there is a difference in the input positions for both patterns and it should be possible for the classifier without feedback to classify them. However, the difference in positions is very small and there could be some confusion if much variation in the shape of the patterns were allowed. If the sharp points of the triangle are detected,and the lack of them detected for the circle,then a much larger shape variation could be tolerated. As before, the only way to detect these features is by some memory in the system.

The structure of the cycle net is shown in Fig. 4.2.2. The learning network contains 48 SLAM-16s which have their clock terminals connected together. There is feedback around the cycle net to provide the necessary memory for features. The input to the cycle net is formed by the input data, the feedback around the cycle net and feedback from the code net.

The major problem with the cycle net is what to teach it. So far, two methods have been tried and neither has been very successful.

In the first method to be investigated, the teach sense terminals are connected to the input data, the net is taught at every step of the input sequence. Hence, the net is taught to output the present position and this is fed back around the net to some of the input terminals (delayed by one step), for the teaching of the next position. Therefore, the cycle net is taught to associate the present step with a mixture of the present and last steps. When, after training the cycle net is tested with an input sequence then, assuming that it works perfectly and synchronises correctly with the input sequences, it should generate the input sequence at its output. Hence the cycle net should behave as a transparent filter for taught input sequences. This method has been investigated for the case of non-sequential input data by Fairhurst and Aleksander(13). However, for the classifier being considered it is necessary for the cycle net to indicate to the code net the sequential distinguishing features of the input sequence, and the transparent filter indicates if an input has been taught but does not process a taught input in any way. Hence, it is not sufficient for the required task and was abandoned. One possible strategy in which this method could be used is to have a separate classifier for each class which is taught to give a high response (i.e., all 1s output) for that class. Then a classification would be indicated by the classifier outputting the highest response.

The second method was to combine the input sequence with the codeword before applying it to the teach terminal of the cycle net. The function of combination that has been used is an exclusive-OR function.

The reasoning behind this teaching method can be demonstrated by a simple example. Consider teaching the cycle net with test data (A) for the two codewords all 0s and all 1s.

When teaching on the circle, the input sequence is exclusive-ORed with 0 (i.e., no change) and the net is taught to output the present step when the input is the present step and the last step as before. When teaching the triangle the input sequence is exclusive-ORed with 1 (i.e., complemented) and the net is taught to output the complement of the present step when the input is the present step and the complement of the last step. If the cycle net, after training, is tested with the circle, assuming that it works perfectly and synchronises with the input sequence, then the input sequence should appear at the output. For the triangle the inverse of the input sequence should appear at the output. The input sequences for the circle and triangle are fairly similar hence, one would expect a large difference between the input sequences for the circle and the complement of the input sequence for the triangle and it should be possible to teach the code net to distinguish between these two sequences.

4.3 INVESTIGATION OF THE PROPERTIES OF THE CLASSIFIER.

4.3.1 Experiment conditions.

All the experiments have some common features which are listed below.

Input Data

The input data for these experiments was A, B or C as described in section 4.1.2. For testing the trained classifier, the training data is reapplied and the responses for the two different classes of input are compared.

Codewords

Two 16-bit codewords have been chosen to represent the two classes, these are octal 000377 and octal 177400. These two codes are the complement of one another and hence, have the maximum Hamming distance (16) between them. They also both have a Hamming distance of 8 with respect to the all-0 codeword which is often used as an initial starting condition.

Learning Networks

For all these experiments, one size of learning network has been used. This consists of 48 SLAM-16 elements for the cycle net (768 bits of active store) and 16 T.R. SLAM-16 elements for the code net (512 bits of active store).

Evaluation of the output response

The output response with respect to a particular codeword is taken as the average Hamming distance between that codeword and the valid output code from the classifier. As the input is sequential one must average the output over many steps (at least one cycle of the input sequence) for a meaningful response. The average number of valid outputs is also noted as this gives a confidence figure for the response. For a deeper measure, the actual code output and the Hamming distance to the different codewords can be recorded at each step.

When evaluating the output response to an input sequence the state transient must be considered. This transient is the number of steps required for the classifier to settle into a stable cycle. To obtain an accurate result it was decided to wait until the initial transient has passed before recording the outputs for an average. It is possible with this program to wait a set number of steps before the output is averaged. The length of the transient is a function of both the structure of the learning network and the nature of the input data. In general, it was found that the initial transient had passed after one cycle of the input pattern had been presented to the classifier. For the data like test data (B), the one identifying feature must be presented at least once to the classifier before a stable cycle can be obtained (assuming that the classifier is able to distinguish between the two classes). For test data (A), in general, the output cycle was established after 30-40 steps. For test data (C) which has very short sequences, the initial transient was also much shorter. In practice, the first 100 outputs for test data (A) and (B) were ignored and the response was averaged over the next 100. For the test data (C) the first 30 outputs were ignored and the next 10 outputs were averaged.

4.3.2 Investigation of the cycle net store.

The classifier using 'Exclusive-OR' teaching described in the previous section was tested with test data (A) and (B) and only worked on a few unconnected occasions (dealt with in section 4.3.3). After training it was seen that the outputs of the cycle net on testing were very similar for the two classes.

The first experiment, described below, was designed to examine the cycle net SLAM stores after training in order to determine primarily if the second class taught to the cycle net was overteaching the first.

Using test data (A) the cycle net is first taught the circle and the number of distinct storage bits of the SLAM stores which are accessed by this teaching are noted. The cycle net is then taught the triangle and all the bits which were originally set to a value by the first teaching and are reset by the second teaching are noted. Hence, the number of elements accessed by teaching a pattern and the amount of overlap when teaching two classes is obtained.

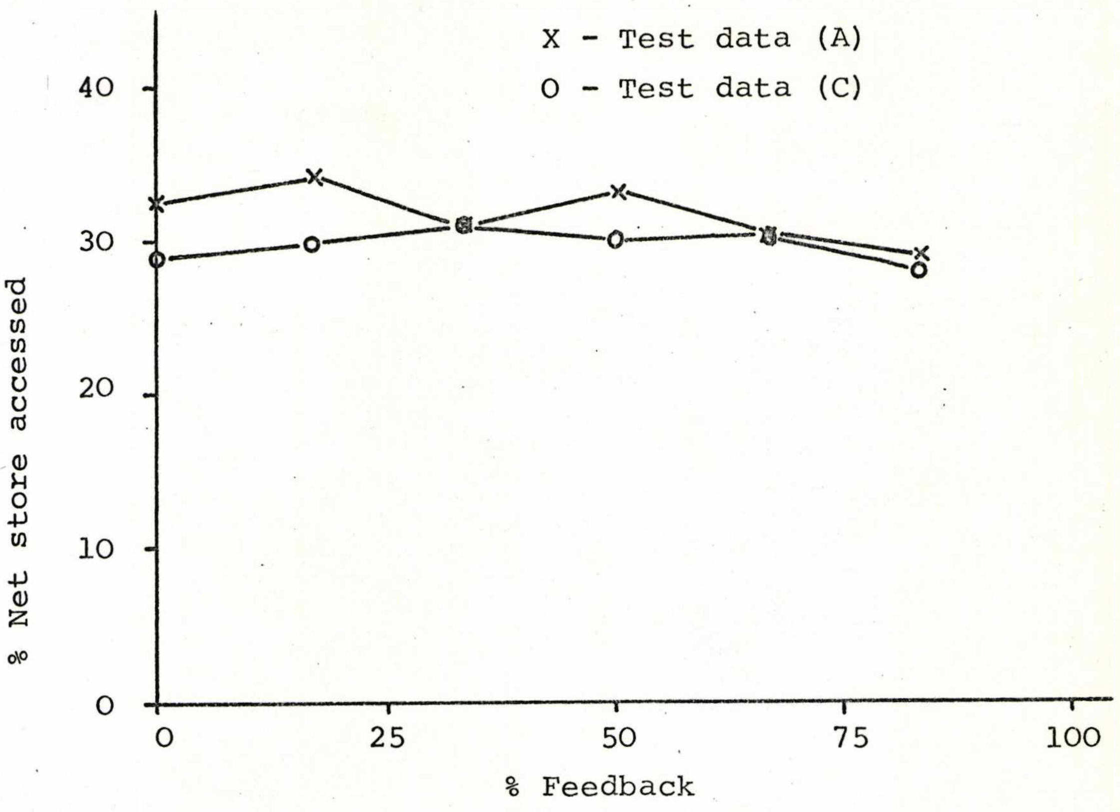
The following is the exact method used to find this number of bits. The cycle net is reset and then taught the circle. The number of distinct bits of the SLAM stores which have become set is recorded. The cycle net is set and then taught the circle again. This time the number of bits which have become reset are recorded. The sum of the two recorded values gives the total number of storage elements which have been accessed by teaching the circle. A similar technique is used to find the number of these bits which are changed on teaching the triangle.

This experiment was repeated with different amounts of feedback around the cycle net and with no feedback from the code net. The graph in Fig. 4.3.1 shows the amount of store accessed when using test data (A) (cycle length ~80 steps, 200 steps taught) and test data (C) (cycle length 8 steps, 40 steps taught). Fig. 4.3.2 shows the amount of relative overlap which was detected for these two sets of data after the second class had been taught.

Fig. 4.3.1 shows that the store penetration (as defined in section 4.2.1) is consistently about 30% whatever the amount of feedback. This is as expected for the feedback is the last input of the sequence exclusive-ORed with a constant codeword and hence should have the same activity (as defined in section 4.2.1) as the input data.

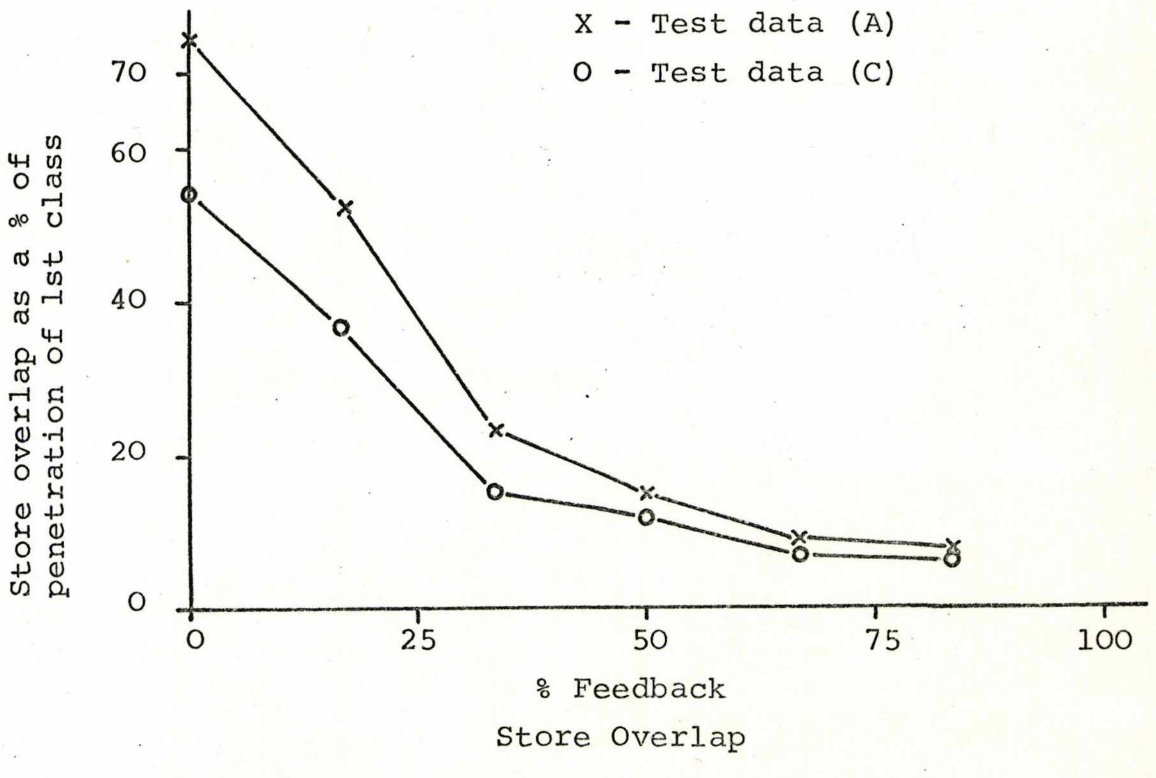
The surprising result from this graph is that almost the same amount of store is accessed by the type (C) test data of 8 steps long as is accessed by the type (A) test data of 80 steps long. This is due mainly to the fact that the range of Hamming distance within both sequencies of patterns is similar.

From Fig. 4.3.2 without feedback there is 74% overlap for test data (A) and 54% overlap for test data (C). The two classes in test data (A) are very close in Hamming distance which would account for the large amount of overlap. As the feedback is increased the amount of overlap decreases. This is because the feedback for the



Store penetration for 1st class

FIGURE 4.3.1



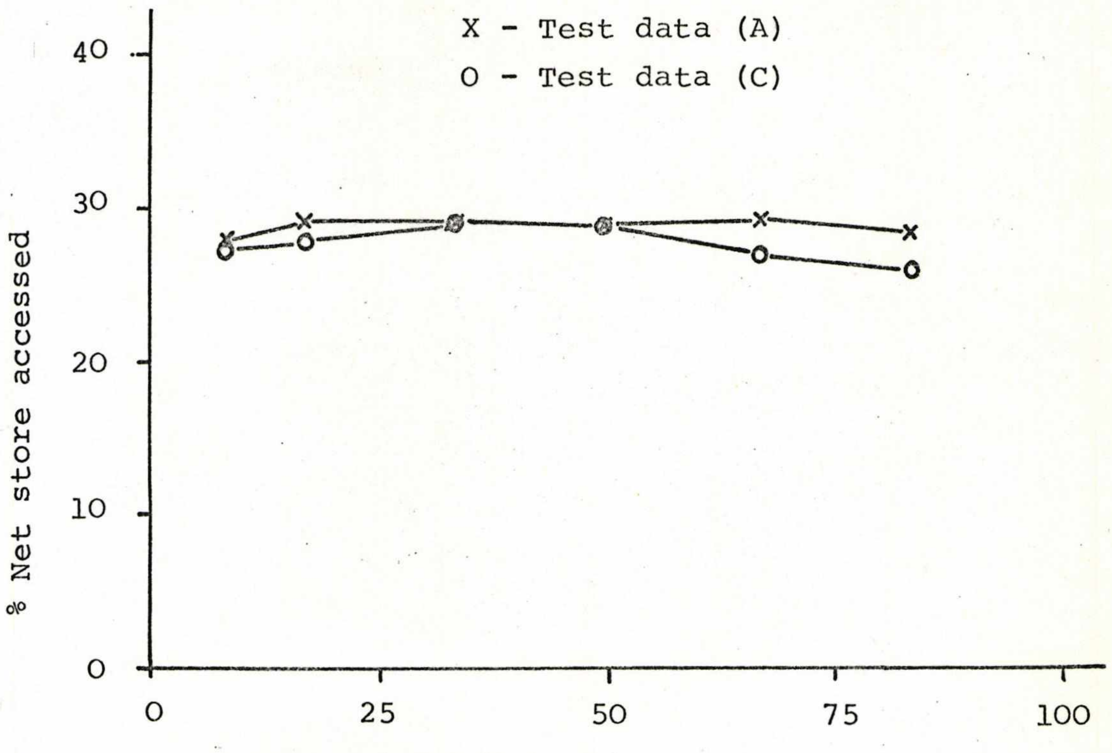
Store Overlap

FIGURE 4.3.2

second class is complemented (by the codeword) with respect to the feedback for the first class and hence, there is a large Hamming distance between them.

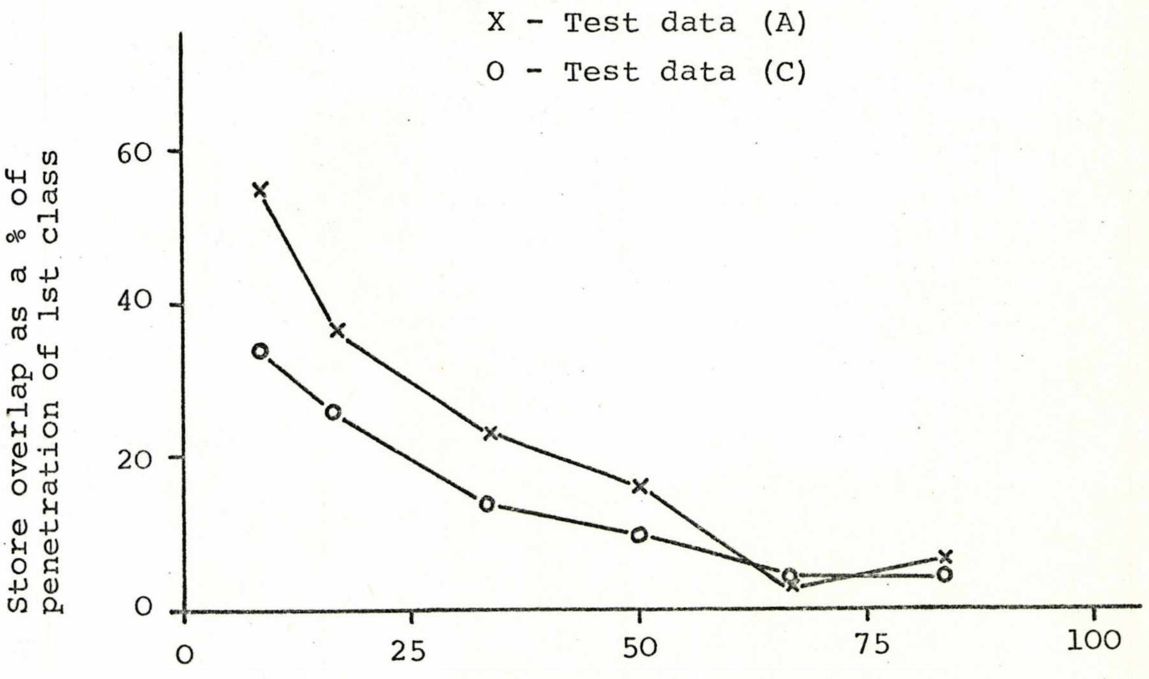
The above experiment was repeated but with 8.3% (1/12th) feedback from the code net i.e., 1/12th of the cycle net had the codeword applied directly to them. The results are shown on Fig. 4.3.3 and Fig. 4.3.4. The amount of store accessed by the first class is about 2% less this time; this is due to the feedback inputs from the code net having a constant value during training. The amount of overlap has been considerably reduced. This is also due to the code net feedback which has a partitioning effect on the cycle net. If the code net feedback had been 25% (i.e., one feedback connection per SLAM element) then the overlap would always be zero.

A measure of how much the stores which were taught for a class were changed by further teaching was obtained (i.e., a measure of how much an input sequence interferes with itself). This was achieved for test data (A) by teaching the cycle net one cycle of the pattern and then to record the stores every 20 steps after that. These stores were then compared with each other and the results obtained are in the table below.



Store penetration (with 8% code net feedback)

FIGURE 4.3.3



Store overlap (with 8% code net feedback)

FIGURE 4.3.4

% bits changed / Number of Steps	100	120	140	160	180	200
100	0					
120	6.8	0				
140	6	6.4	0			
160	6	8	4	0		
180	5.6	9.2	7.6	4.8	0	
200	2	6.4	4.8	5.6	5.2	0

For this experiment the feedback was 50%. However, as with the amount of store penetration, one would expect the above figures to be independent of the amount of cycle net feedback. The amount of store penetration for this case is 33%, the figures in the table are expressed as a percentage with respect to this penetration. Hence, the maximum observed change in the stores caused by the teaching of one pattern interacting with itself is 9.2% of the total store accessed.

To summarise, the analysis of the cycle net stores for both type (A) and type (C) test data has shown the following:

1. About 33% store penetration occurs on teaching the first pattern to the cycle net. This penetration is independent of the amount of cycle net feedback but is reduced by the addition of code net feedback. (In the limit with 100% code net feedback the store penetration would be 6.25%).

2. Without feedback there is more than 50% overlap between the two classes. This is reduced progressively as cycle net feedback is added.

3. Both store penetration and store overlap (expressed as a fraction of store penetration) are reduced by code net feedback.

4. For test data (A) about 10% of the store penetrated is taught to output both 1 and 0 during the cycle of the input pattern.

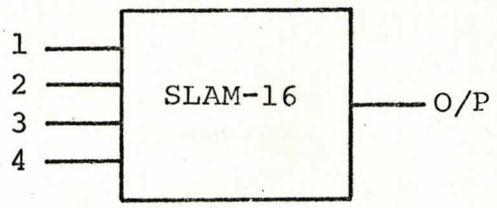
The main point that this experiment shows is that for the useful range of feedback (defined in section 4.3.4) there is between 15-70% store overlap between two classes and in the order of 10% store overlap within class. What is not shown by this experiment is the frequency of occurrence that these overlapped elements are accessed. The properties of the exclusive-OR method of teaching will be discussed further in section 4.3.4.

4.3.3 Investigation of the code net store.

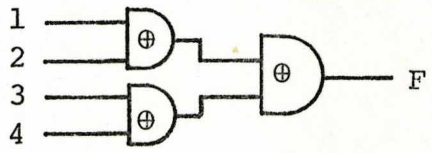
The experiment in this section is designed to demonstrate some of the properties of the code net and the effect of feedback from the output of the code net on the classifier. As the teaching method for the cycle net was not very effective the code net was tested with fixed, untaught stores for the cycle net as well as the taught cycle net. The SLAM element may have any predefined truth table loaded into its stores so that it performs a particular function. The functions that were used in the cycle net are shown in Fig. 4.3.5. The probability of a 1 being output for a random input is also shown (Pr).

The code net consists of T.R. SLAM-16s hence, after teaching, their stores may be easily analysed. The results are shown in Figs. 4.3.6 to 4.3.8. The elements of the T.R. SLAMs have been divided into three types; valid, untaught and overtaught. These values have been plotted in the graphs. The different type of cycle net stores which have been used are set along the x axis in order of descending penetration of the T.R. SLAM store. All the experiments were those with the type (B) test data and with several values of feedback from both the cycle and the code net.

After each store was recorded the classifier was tested with the training set again and the average response after 200 steps was recorded. When there was no feedback from the code net the average response was in nearly all cases 100% correct but the average number of valid outputs



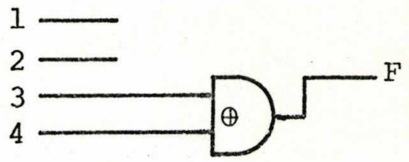
1 Exclusive OR



$F = (1 \oplus 2) \oplus (3 \oplus 4)$

$P_r = 0.5$

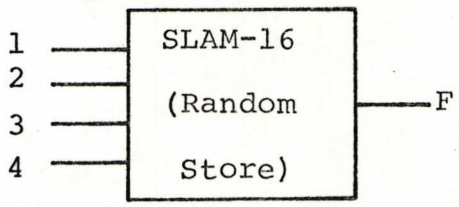
2 2 I/P Exclusive OR



$F = 3 \oplus 4$

$P_r = 0.5$

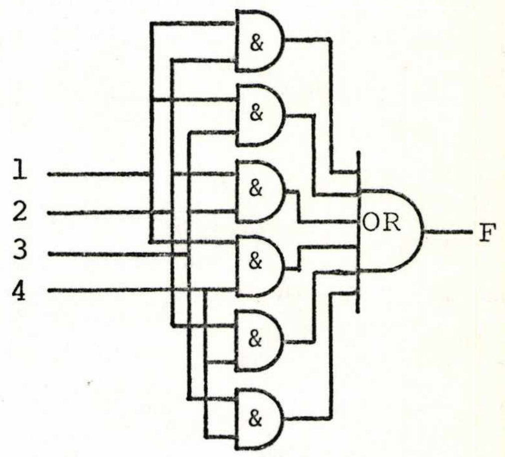
3 Random



(Any function is possible)

Max-liklihood $P_r = 0.5$

4 Majority Gate

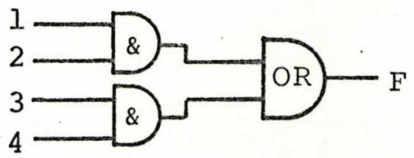


$F = 1$ if $N \geqslant 2$

where N is the number of I/P which are 1

$P_r = 0.69$

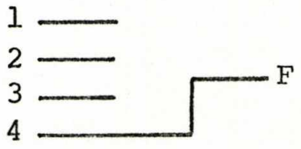
5 AND-OR



$F = 1.2 + 3.4$

$P_r = 0.5$

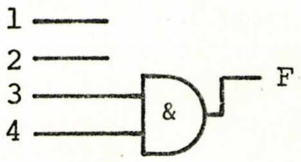
6 No gates

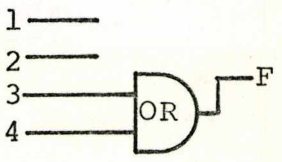


$F = 4$

$P_r = 0.5$

7 2 I/P AND or OR





$F = 3.4$ or $F = 3+4$

$P_r = 0.25$ or $P_r = 0.25$

FIGURE 4.3.5

	% Feedback	
	Cycle	Code
+	8.3	O
O	O	8.3
□	8.3	8.3
Δ	25	8.3
X	8.3	16.6

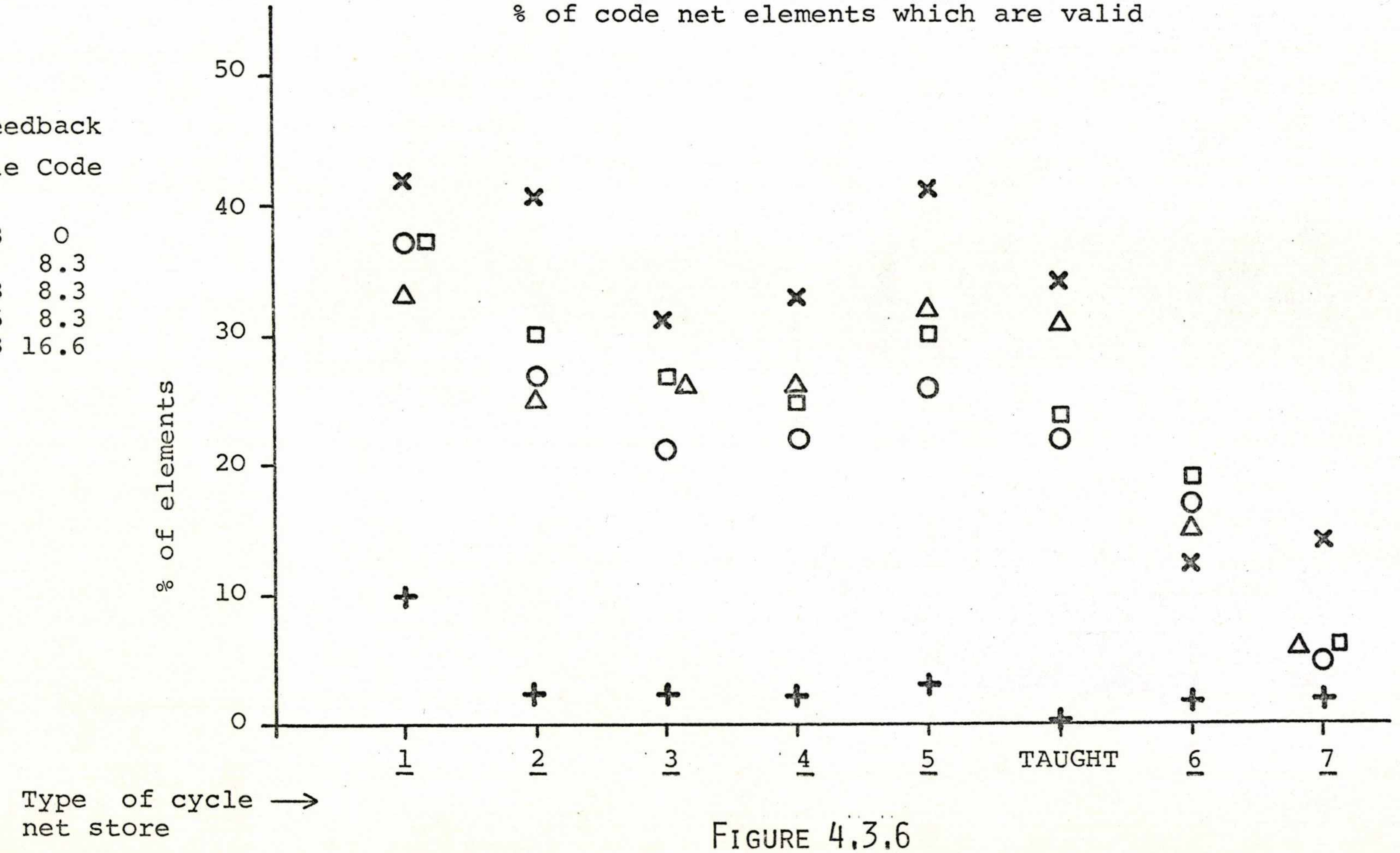
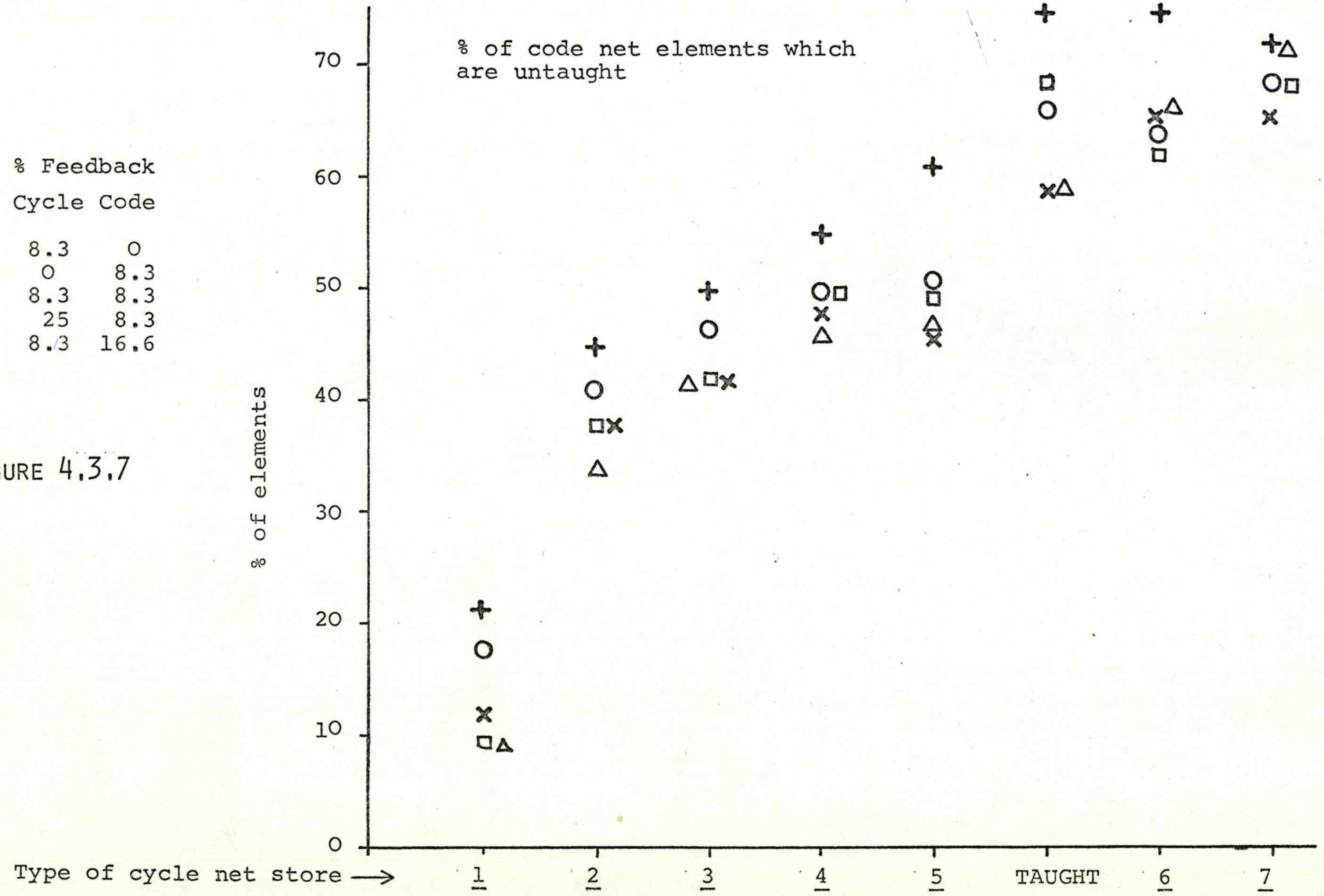


FIGURE 4.3.6

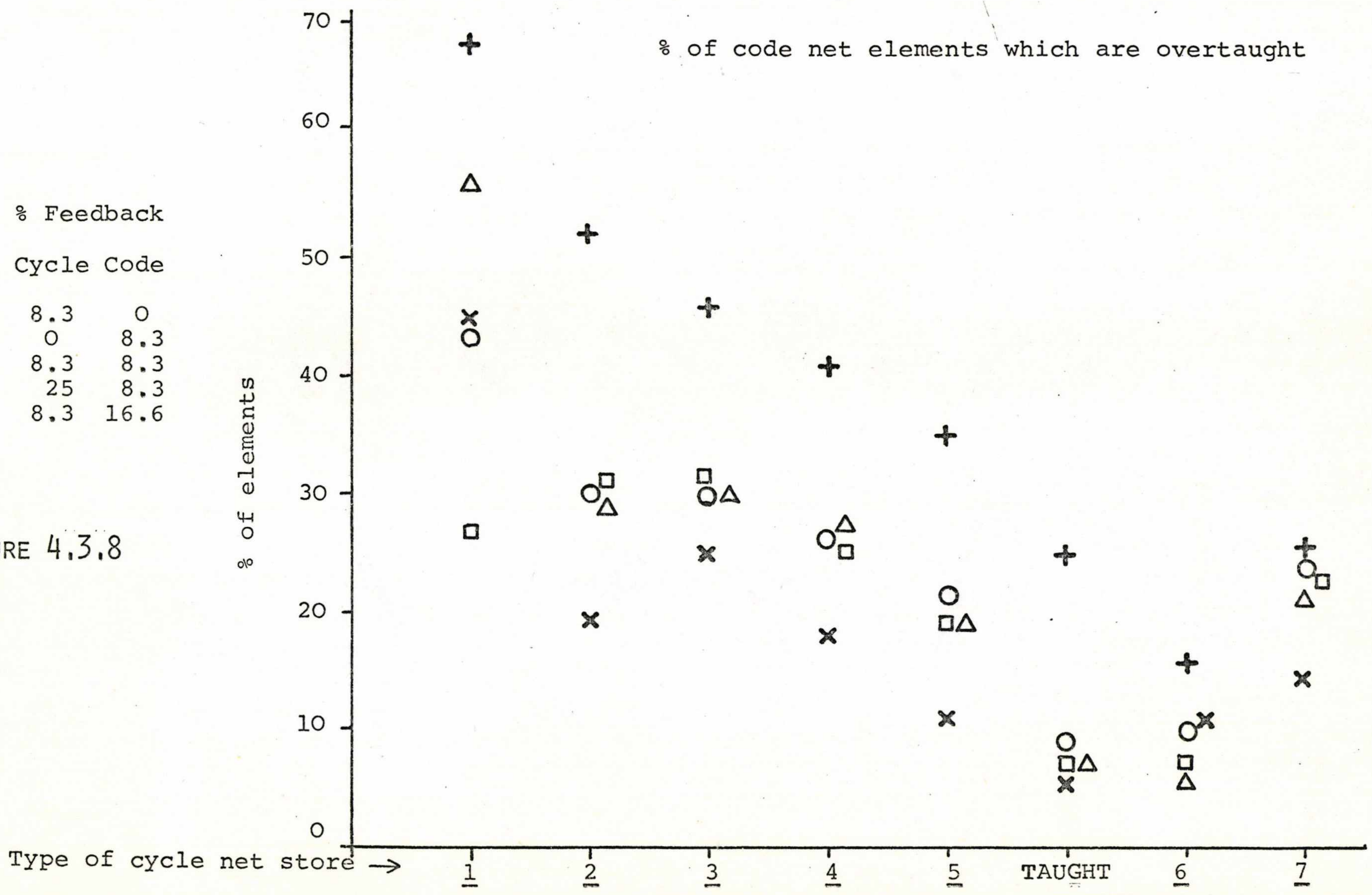
	% Feedback	
	Cycle	Code
+	8.3	0
O	0	8.3
□	8.3	8.3
Δ	25	8.3
X	8.3	16.6

FIGURE 4.3.7



	% Feedback	
	Cycle	Code
+	8.3	0
O	0	8.3
□	8.3	8.3
Δ	25	8.3
X	8.3	16.6

FIGURE 4.3.8



was always less than 2%, which implies that the classifier in this form is not suitable for the task. This will be discussed further in the next section.

With feedback from the code net there were always many more valid outputs (the lowest observed average valid output was 9%). However, the correct response was only obtained on a few occasions. These are listed below.

	% cycle net feed-back	% code net feed-back	cycle net store	Average Responses To ⊟ % data response	valid % re-sponse	To □ % data response	valid % re-sponse
1	8.3	16.6	(2) 2I/P x OR gate	98	58	70	58
2	8.3	16.6	(3) Random Store	67	38	56	36
3	25	8.3	(4) Majority gates	99	27	54	27
4	8.3	16.6	(5) AND-OR gates	98	59	97	52
5*	25	8.3	(5) AND-OR gates	67	19	52	19
6*	8.3	16.6	(1) x OR gates	61	48	76	40
7*	25	8.3	(2) 2I/P x OR gates	65	42	51	40

Each test was repeated 4 times with different initial starting values for the feedback. The results above indicated by * gave the response shown for some starting values but gave an incorrect response for others. (A correct response for the two-class case, which has been investigated, occurs if the data response is 50% for the correct class.)

In the results, one example of 0% cycle net feedback is given (with 8.3% code net feedback). Results have also been obtained for 0 and 16.6% code net feedback and they are very similar to the example plotted. The example shows that there are very few valid elements and this means that only part of the correct codeword can be generated and meaningful classifications cannot be made. Although the classifier could usually classify the training data with about 1% confidence, when it was tested with other similar data it usually gave a zero output.

4.3.4 Observations from the experiments.

1. Code net feedback.

The different amounts of code net feedback which were used to test the classifier are 0, 8.3, 16.6 and 25%. Considering test data (B) with no code net feedback, there is no definite latching into one state when the one defining feature occurs and the activity of the outputs of the cycle net during the cycle were always very similar for both classes. For 25% feedback the random map used implies that every SLAM in the cycle net receives one code net feedback connection; then the system tended to latch into one of the two classes and after the first few steps of the input sequence the classifier was set in one class and never changed. This class was often determined by the initial feedback pattern and was rarely determined by the input sequence. For test data (A) similar effects were observed but these were not as marked as for test data (B). This is due to the fact that with the test data (A) there is a 'clue' to which class the input belongs to whenever the sequence is started whereas with test data (B) there is only one part of the cycle which defines the class.

If 25% feedback is used the cycle net is partitioned by the codeword on teaching into two similar learning networks of SLAM-8s. Hence, no overlap on teaching occurs. However, one would expect that once one codeword has stabilised (i.e., one network has been selected) that a lot of change at the input would be required to make a change to the other class. This explains why the test data (A) and (B) were unable to change the classification once it had been established. Further increase in code net feedback (for the two-class case) will make matters worse, especially when the cycle net is taught as many 4-tuple input patterns which may easily occur during testing cannot occur when training.

Hence, for the two-class case the code net feedback must be less than 25%, both 8.3% and 16.6% feedback were found to be useful.

2. Cycle net feedback.

To define useful limits for the cycle net feedback is more difficult than for the code net as the nature of this feedback depends on the functions of the SLAM stores or the method of teaching the cycle net. Values of cycle net feedback between 0 and 50% were investigated. In general, small values of cycle net feedback were the useful. For test data (B) the only correct results were obtained when values between 8.3% and 25% feedback were used. For test data (A) where this type of feedback is not essential, the best results were obtained with feedback in the range 0 to 25%. However, some correct results were obtained with more feedback.

Once more than 25% feedback with 4 input SLAMs is used the system becomes potentially unstable as one output may affect two others on the next cycle and four others on the next, etc. Therefore, although the classifier may work with more than 25% feedback, one must remember that it is potentially unstable.

3. The performance of the classifier.

The classifier did not classify the real data reliably over a range of values for feedback for any of the stores of the cycle net used. The behaviour of the classifier was rather erratic in that it might work for two values of feedback but not for a mid value between them. This may be due in part to the small size of the learning network used. What is clear however, is that when the cycle net is tested with the training set it does not go into the cycles it was originally taught and the cycles it does go into are usually very similar for both classes. Further developments of the classifier are considered in the Conclusion (section 7.2).

4. The performance of the exclusive-OR method of teaching.

When compared with set stores in the code net store experiment the exclusive-OR method of teaching does not behave very well. As far as activity at the outputs is concerned (measured by the penetration of the code net elements) it is lower than any of the set stores for four input cycle net elements including the random elements (which are the initial states of the stores before exclusive-OR teaching). However, the number of valid elements in the code net created by the taught cycle net is similar to the number created by the set stores (i.e., the ratio of valid to non-valid elements is greater for the taught case). From this one might expect a better performance from the taught cycle net but this has not been observed. The poor performance may be due to the store overlap which occurs when teaching the cycle net (see section 4.3.2).

Two methods have been tried to improve the performance of the exclusive-OR teaching method. The first was only to teach part of the learning network and to leave the rest random, and the second was to teach the cycle net for a fraction of the time that the input is presented. However, neither of these methods produced a noticeable improvement.

4.4 SHIFT REGISTER CLASSIFIER.

4.4.1 Structure.

The general structure of the shift register classifier is shown in Fig. 4.4.1. The 4-bit tracking movements are fed directly into a 4-bit wide shift register which has a maximum length of 64 steps. The contents of the shift register are randomly connected to the input of a set of learning networks. There are N learning networks with their inputs connected in parallel. Each network represents one class of patterns hence, N classes may be taught. The first learning network is shown in more detail in Fig. 4.4.1. All the teach sense inputs are set to 1. The outputs are summed to obtain a response.

Initially, all the SLAM stores are set to 0 (i.e., will give a 0 response for all inputs). During training the learning network of the relevant class is clocked when the input sequence is applied. When the input sequence is used to test the classifier this learning network will output a maximum response. In general, a classification is made by comparing the average responses from all the learning networks, and the highest average response decides the class.

4.4.2 Peformance of the classifier.

Tests to evaluate the performance and properties of this classifier have been conducted using test data (**A**).

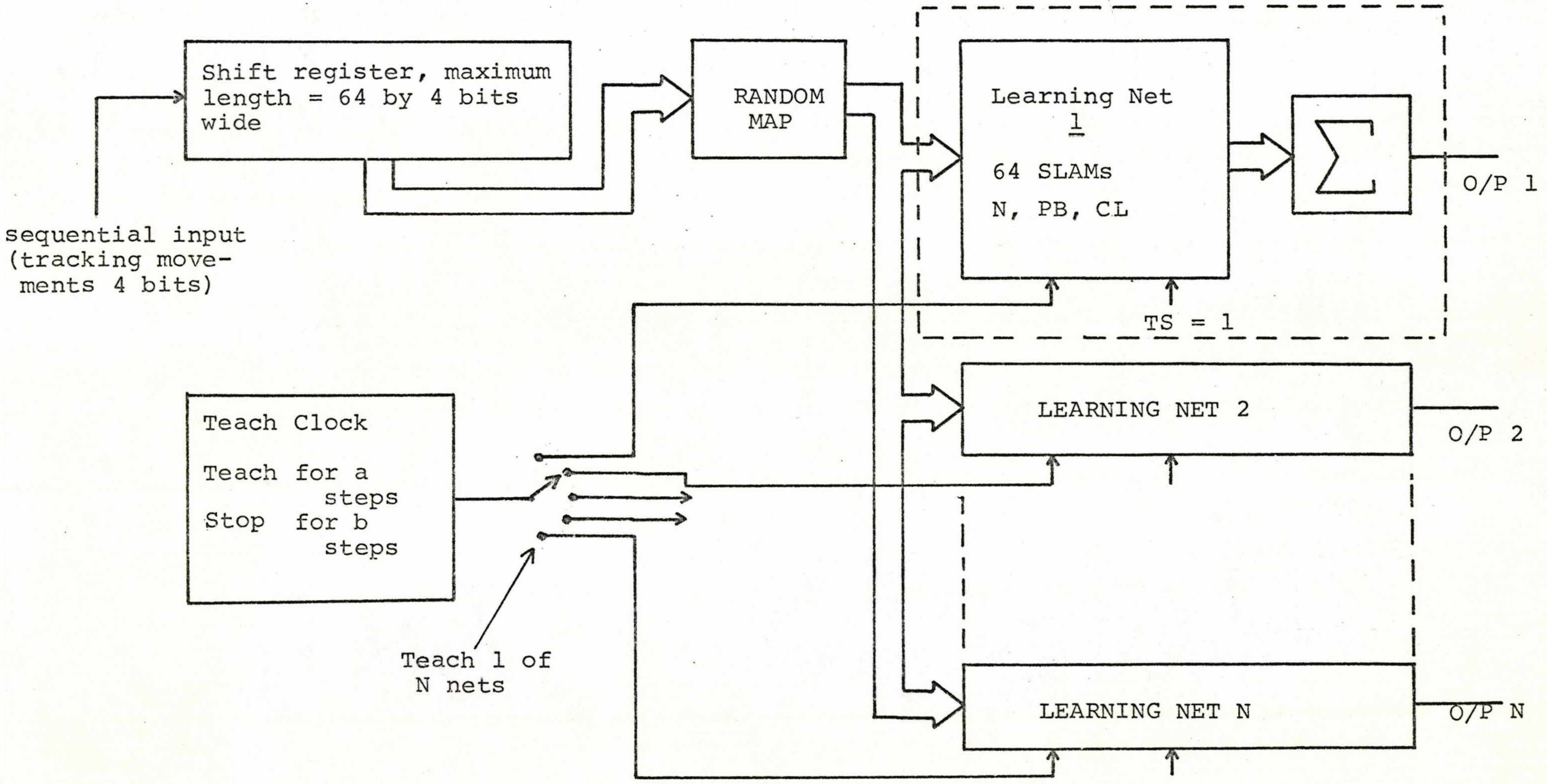


FIGURE 4.4.1 SHIFT REGISTER CLASSIFIER

Each test involves two learning networks. One learning network is taught to output a high response when tracking data for the circle is input and the other is taught to give a high response for the triangle. Four different types of SLAM learning networks have been tested in this way, these are:-

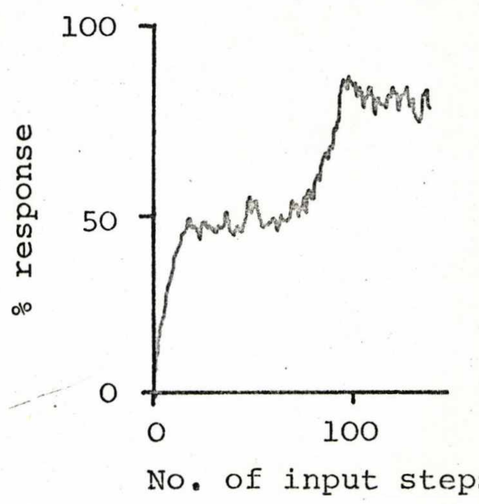
1. Normal SLAM learning network, 64 SLAM-16s.
2. Probability SLAM learning network (probability = 1/16).
3. Probability SLAM learning network (probability = 1/4).
4. 4-layer CL SLAM-16 learning network, 256 SLAM-16s.

Further details of the construction of these learning networks are given in Chapter 5.

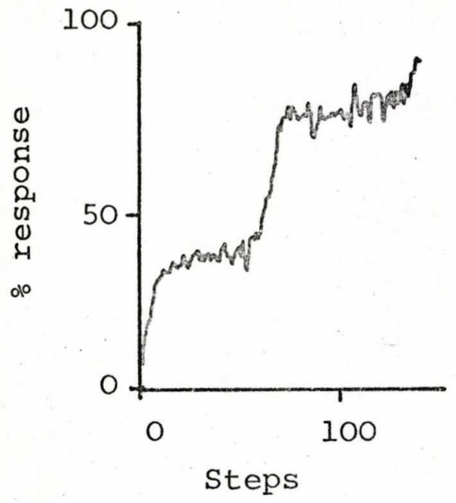
Several different sizes of shift register were tested and the longest (64 steps) produced the best results. The results given in the table below were the best obtained. 150 steps of each class were taught to the learning networks and a subsequent 200 steps of each class was used to test them.

	Average % Response from O Learning Network.	Average % Response from Δ Learning Network
1. Normal SLAM Net		
Test with O	98.9	95.0
Test with Δ	90.2	98.9
2. Probability Net		
Prob. 1/16		
Test with O	63.2	54.0
Test with Δ	57.0	64.7
3. Probability Net		
Prob. 4/16		
Test with O	91.9	83.4
Test with Δ	84.2	93.6
4. CL SLAM Net		
Test with O	81.1	70.3
Test with Δ	72.9	83.1

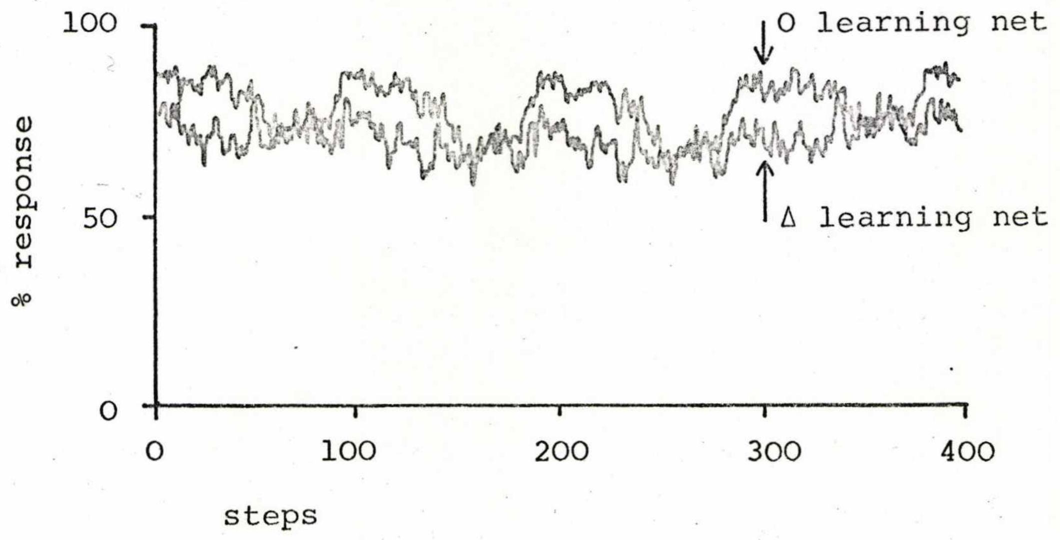
In all the above cases the correct class could be detected by the highest response. An illustration of how the response at the output appears is shown in Fig. 4.4.2. This example is for the CL SLAM learning network structure. Fig. 4.4.2(a) shows how the output response varies when the circle is being taught and (b) when the triangle is taught. The responses for both learning networks when tested with



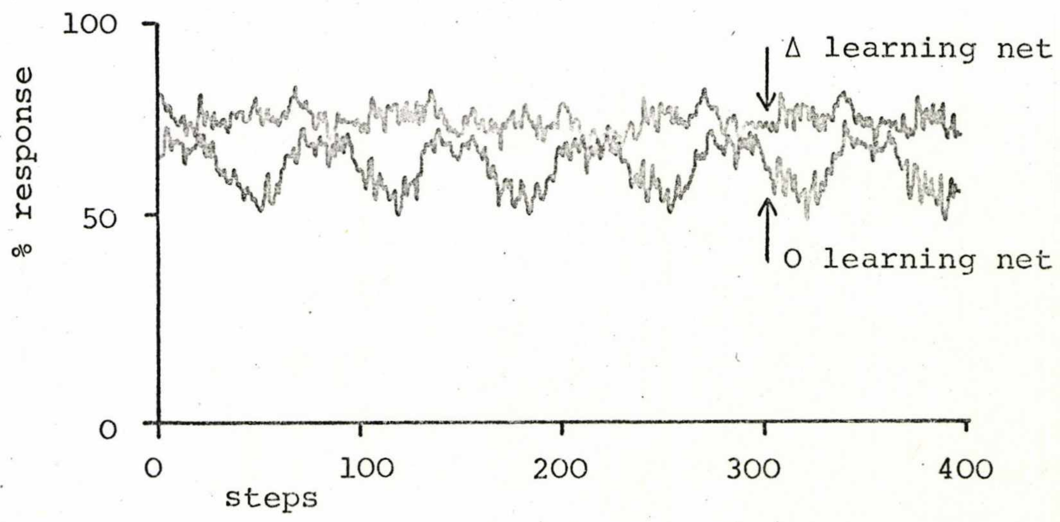
(a) training O



(b) training Δ



(c) when tested with O



(d) when tested with Δ

FIGURE 4.4.2

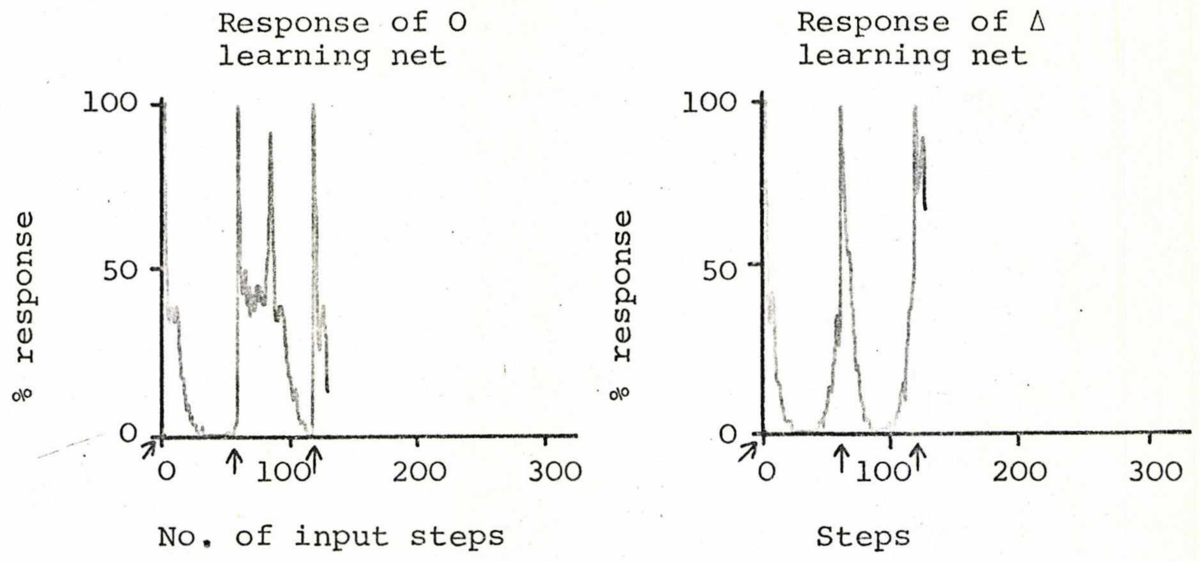
the circle are shown in (c) and when tested with the triangle are shown in (d).

The shift register method has two fundamental problems, size invariance and overteaching.

The size invariance occurs because the shift register is of a fixed length whereas the length of the input sequence depends on the size of the input pattern. Hence, for different size input patterns, the features will occur at different relative positions in the shift register.

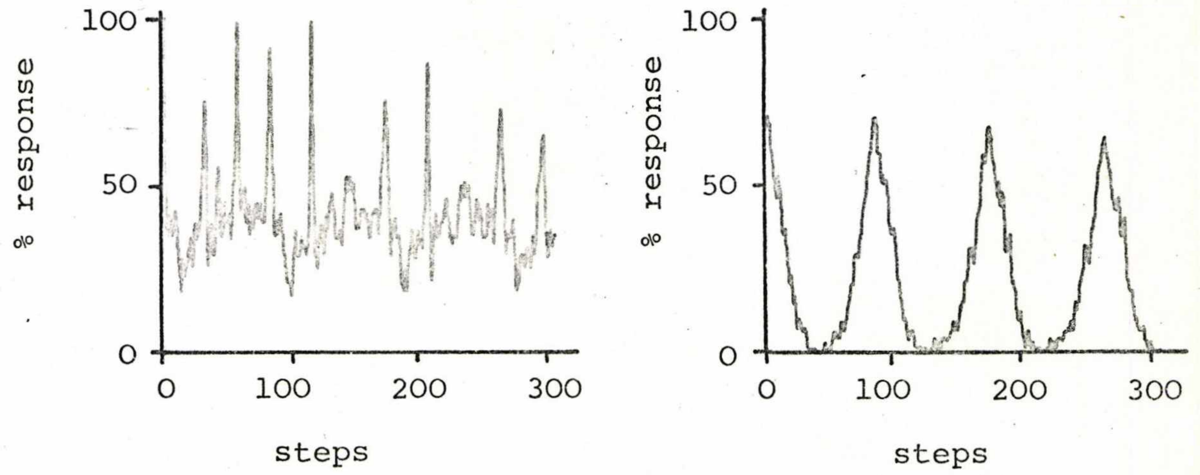
Overteaching occurs because many very different patterns occur in the shift register and they are all taught to the learning network. The normal SLAM network gives very high responses to all input sequences after being taught only one of a class. This overteaching is reduced by the net organisation in the other learning networks tested. However, if several examples of a class are taught, then these other learning networks also become saturated.

One possible method of overcoming the overteaching problem is to only teach the learning network when a new, completely untaught, set of data is in **the shift register**(e.g., once every 64 steps for a 64 step shift register). An example of how the classifier behaves in this situation is shown in Fig. 4.4.3. The shift register is of length 64 and the learning network is the 64 normal SLAM type. The learning network is taught three times during the 150 steps for teaching (i.e. once every 64 steps). In this case, the

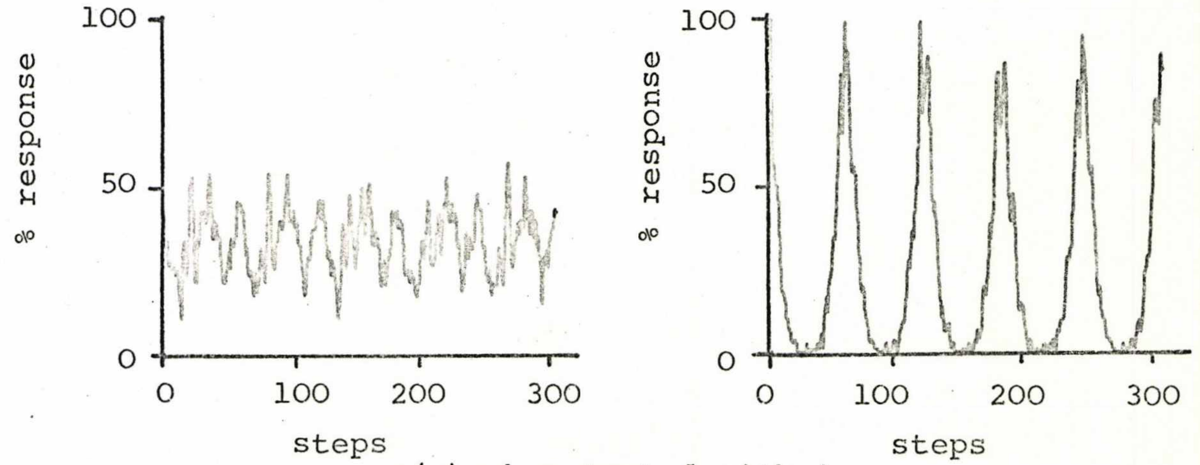


(a) when trained

↑ indicates when the net is taught



(b) when tested with O



(c) when tested with Δ

FIGURE 4.4.3

correct classification could be obtained by detecting the highest peaks in the response. In general, however, this method has produced many more problems e.g.

1. When classifying, the frequency of occurrence of the peaks should also be considered and this is complicated by the fact that the input sequence frequency 'beats' with the teaching frequency. This can be seen in Fig. 4.4.3.

2. When teaching a set of input patterns, they should all be synchronised i.e., teaching should only occur at the same relative positions for each input pattern.

CHAPTER 5
THE LEARNING ELEMENTS AND NETWORKS

In this chapter the learning elements used in the learning networks are considered in detail. The element from which the others have been derived is the SLAM-16 and this is described in section 5.1.

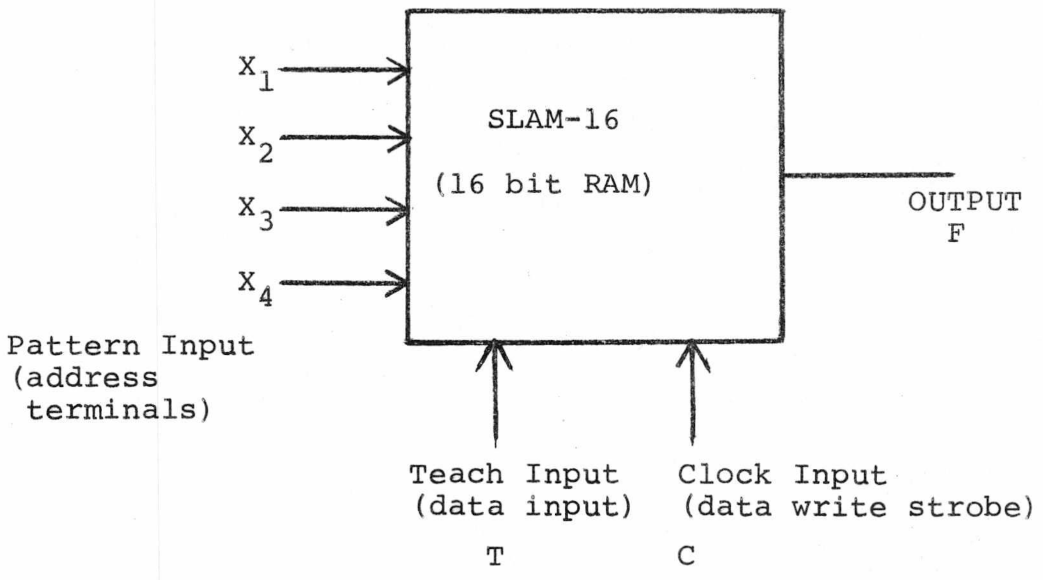
The way in which this and alternative elements may be organised to distinguish between two classes is considered and, in section 5.5, the different elements are compared.

5.1 THE SLAM ELEMENT.

5.1.1 The SLAM-16.

A SLAM (Stored Logic Adaptive Microcircuit) is a random access memory (RAM) used as an adaptive logic gate. A SLAM-16, for example, is a 16-bit RAM and therefore has four address inputs, which are the data inputs for that element. If a 1 is written in to the memory for a particular 4-bit input pattern, then this element will produce a 1 at the output if this input pattern reoccurs. It can also be taught to associate a 0 output for a particular input pattern. The writing input to the memory is referred to as the teach terminal. An illustration of a SLAM-16 showing its terminal functions is shown in Fig. 5.1.1.

A collection of these elements has been connected



X_1
X_2
X_3
X_4
SLAM-16
(16 bit RAM)
OUTPUT F
Pattern Input (address terminals)
Teach Input (data input) T
Clock Input (data write strobe) C

In the general case the n input SLAM contains 2^n binary storage elements, $\phi_1 - \phi_{2^n-1}$ and is called a SLAM-(2^n).

The function of the element is defined by:

$$F = \bigcup_{j=0}^{j=2^n-1} \phi_j \langle x_j \rangle$$

where $\langle x_j \rangle$ is the jth minterm of the inputs $X_1 - X_n$.

The teaching function is defined by:

$$\phi_j{}' = \langle x_j \rangle . T . C + \phi_j . \overline{C . \langle x_j \rangle}$$

where $\phi_j{}'$ is the next state of ϕ_j

T is the Teach Sense Input value

C is the Teach Clock Input value.

FIGURE 5.1.1

to form a learning network. Usually the inputs of the elements are randomly distributed in an input 'retina' so that each SLAM samples a random 4-tuple of this total input. The learning network can be taught to associate any one pattern applied to the teach sense terminals with a particular pattern in the input space. Hence, after teaching, if that input pattern is reapplied to the learning network the taught pattern will be generated at the output.

A learning network of these elements is able to generalise, in that if it is taught to output a particular response for a class of input patterns then it will output a similar response when tested with an unseen pattern which is similar to the training patterns in Hamming distance. This has been dealt with in detail (1) for the case where the learning network is initially reset and is then taught to output all 1's for a set of input patterns.

Throughout this project only SLAM-16s have been considered. The effect of varying the number of inputs of the elements for the tasks which have been considered here is not known. It is almost certain that four inputs per SLAM is not the optimum and it must be understood that this is a constraint on the system. Further work could be carried out on trying to discover the optimum for this task. This is considered to be outside the scope of the present thesis. The SLAM-16 was used as the hardware learning element in the 'Minerva' learning machine (see section 5.2) and it is also convenient to use one 16-bit word of the computer store in the simulation of such an element.

5.1.2 The 2-class learning network.

A task which is often required of learning networks is to distinguish between two classes of inputs. (e.g. the tracking system learning modules, see section 3.2.1).

This is usually achieved by using two learning networks, one for each class. These learning networks are initially reset and are taught to output all 1's for their particular class. The class of an unseen pattern is decided by the learning network with the highest response. The performance of such a learning system, if the same input mapping is used for both learning networks, is similar in basic structure and performance to a single learning network of T.R.SLAMs which is described in detail in section 5.5.4.

Another, usually less successful, method is to use one learning network. This is taught to give an output of all 1's for one class and all 0's for the other, and the response to a test pattern is obtained by summing the number of 1's appearing at the output. This response can be compared with a threshold (usually set at 50%) to decide the class of the input.

This network may be realised by SLAM-16s. All the teach sense terminals are connected together so that all 1's or all 0's can easily be presented to them and all the teach clock terminals are connected together so that all the SLAMs may be simultaneously taught. This network will always reproduce the response for the last pattern taught. However, especially if the Hamming distance between the two input classes is small, the learning network will be unable to give the correct maximum or minimum response to patterns used for training but may give a response great enough for the two classes to be distinguished.

5.2 MINERVA.

Minerva is a hardware machine which contains 1280 SLAM-16s. These SLAMs are organised in two layers 1048 in the first which have their outputs connected to 256 in the second. In this project, the second layer has not been used and no further reference will be made to it.

Minerva is organised into 256 cards each containing 4 SLAM-16s. One card at a time is accessed by the computer. This can be done by outputting its address directly from the computer, or by incrementing a hardware counter in Minerva through a consecutive stack of cards. One of these cards is shown in Fig. 5.2.1. The input is a 16-bit word from the computer, common teach-sense and teach-clock terminals are available to the computer (via pulses on output control lines) and a 4-bit output from the card is input to the computer.

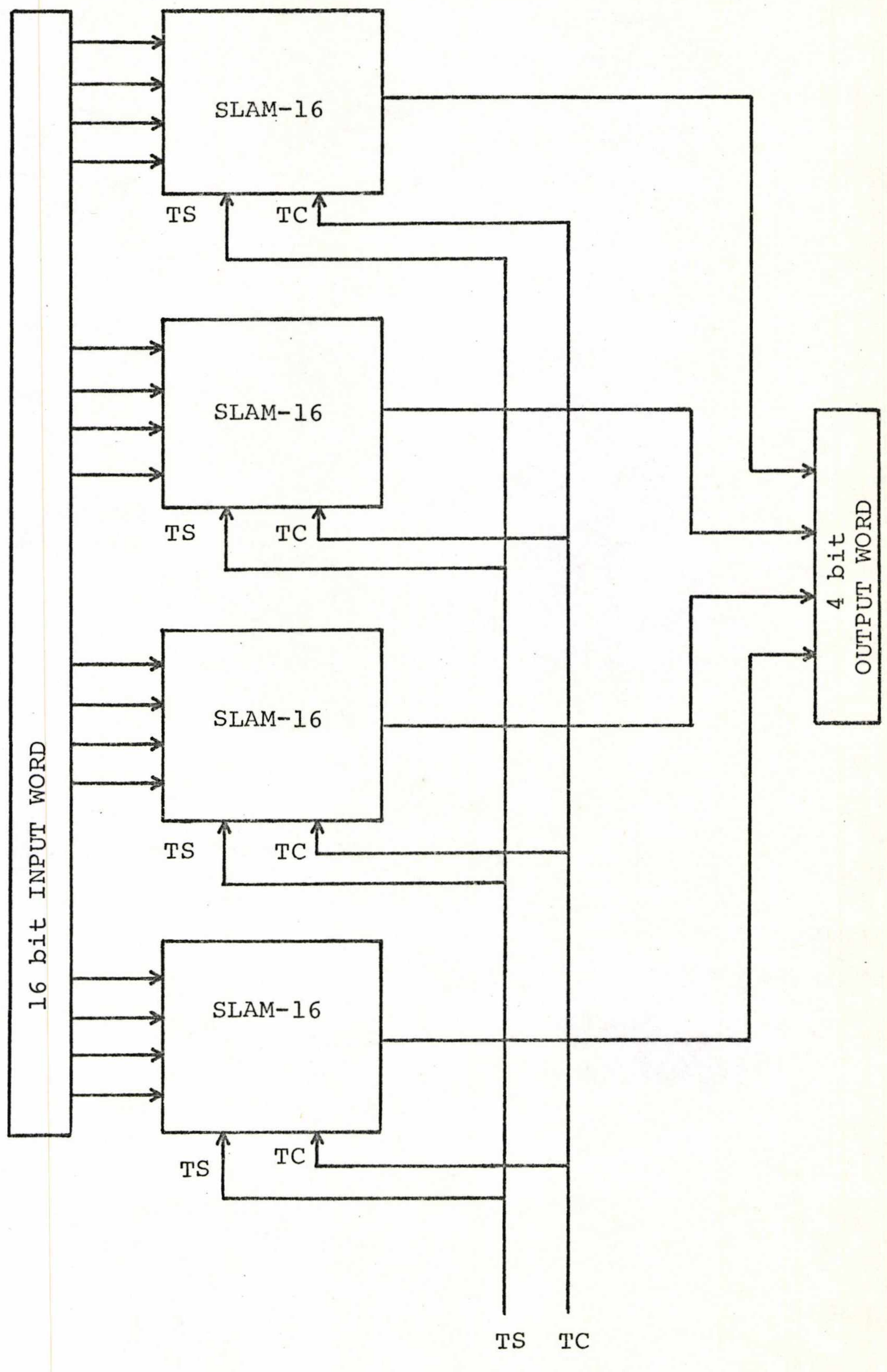


FIGURE 5.2.1 MINERVA CARD ORGANISATION

In all possible cases, experiments have been conducted both with Minerva and by computer simulation of SLAMs. The advantage of Minerva is that it is about 50 times quicker than the simulation. The main disadvantage with Minerva is that the teach sense and teach clock terminals are common for sets of 4 SLAMs and for many structures it is desirable to access these terminals separately. Minerva is being further developed and in the future separate access to these terminals will be possible.

5.3 PROBABILITY LEARNING NETWORKS.

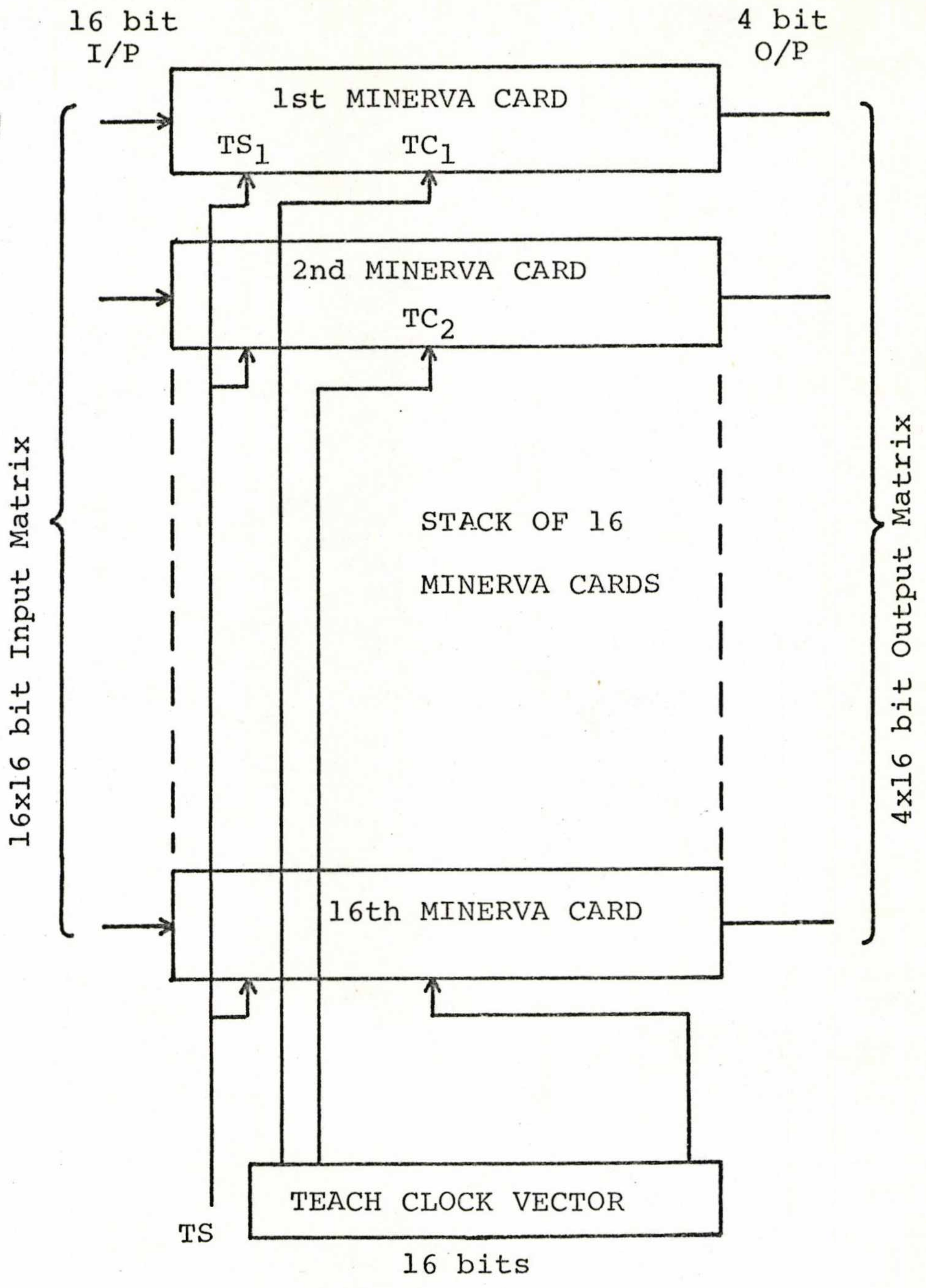
The probability learning network is a structure of SLAM-16s in which an attempt has been made to reduce the effect of the last pattern taught. This structure is primarily intended for cases when the learning network is taught to output a response as described in section 5.1.2. The problem with teaching all the SLAMs for one class is that when the two classes are similar then many of the 4-tuple samples input to the SLAMs will be common to both classes and information in these SLAMs will be continuously overwritten as the alternate classes are taught. Hence, there will be a bias towards the last class taught.

To overcome this bias, the probability SLAM network is used, in which only a randomly chosen fraction of the network is taught for each input pattern. This has been designed to be realised with Minerva or by simulation. A standard size learning network of 64 SLAM-16s has been chosen for this structure, although in practice, any multiple of 16 SLAM-16s may be used in the same way. The structure of this learning network as realised by 16 Minerva cards is shown in Fig. 5.3.1. The input matrix is 16x16 bits (i.e., the standard size for the software system). The output matrix is 4x16 bits which may be summed to obtain a response as before. All the teach sense terminals are connected together and hence they can all be set at 0 or 1 depending on which class is being taught. The teach clock terminals of the different cards are connected to different bits of a 16-bit teach clock vector. Only the cards which are associated with bits in the teach clock vector that are set to 1 will be taught.

The teach clock vector is generated by a random number generator which is designed to set N bits randomly in a 16-bit word. Where N is a preset value ($1 \leq N \leq 16$) and determines the fraction of the learning network to be taught when it is clocked.

Hence, if N=4, for example, a randomly chosen quarter of all the cards will be taught at a time, if N=1 then 1/16th of the cards will be taught at a time.

The advantage of this learning network is that if N=1 for example, then only 1/16th of the SLAMs will be taught to the last class taught. Hence, only in the order of 1/16th of the SLAMs common to both classes will



ORGANISATION OF A PROBABILITY
SLAM LEARNING NETWORK

FIGURE 5.3.1

be taught to output for the 'last class taught'.

The main disadvantage of this learning network is that if n=1, for example, only 1/16th of each input pattern will be taught to the learning network and the rest will be ignored.

One method of teaching this learning network is to start with n=16 so that the learning network is rapidly taught and then, as teaching progresses, decrease n slowly until n=0. In this way the effect of the last pattern taught is removed. This method of teaching with 'aging' has been investigated elsewhere(14).

The performance of this learning network is investigated in section 5.6.

5.4 THE CUMULATIVE LEARNING SLAM ELEMENT.

5.4.1 Structure of the element.

The Cumulative Learning SLAM, or CL SLAM, has been developed to be sensitive, during training, to occurrence frequencies of n-tuples and, in this way, to overcome partly the bias towards the more recently taught patterns. The CL SLAM has a numerical output the value of which is related to the frequency of occurrence of the input pattern during training.

The CL SLAM contains a stack of normal SLAMs. Fig. 5.4.1 shows a 4 output CL SLAM-16. Hence, 4 bits of output information are associated with every 4-bit input pattern. The teach clock terminals are connected together to form a single teach clock terminal for the element. The four outputs from the element represent a 4-bit binary number. The output number is fed to an adder unit which is connected to the teach sense terminals of the SLAMs. The teach sense input for the element is also connected to the adder unit which works on the following principle. When the teach sense terminal is 1 the output is made 1 higher than the input. When the teach sense terminal is 0 the output is made 1 lower than the input. Saturation occurs at 15 when ascending and 0 when descending. (When this element is used for the 2 class learning network saturation occurs at 1 when descending so that a mid value '8' exists which corresponds to a 50% output).

Therefore, when an input pattern is taught to the CL SLAM and the teach sense terminal is 1 then the SLAM will be taught to output a number one higher than it did for that pattern before teaching i.e., the number associated with that input pattern is incremented by one.

Hence, if the teach sense terminal is held at 1, the output number represents the number of times that the input pattern has been taught until, for a 4 output element, the output reaches 15. Then the system saturates and continues to output 15 unless the element is taught with the teach sense terminal at 0 and then the element

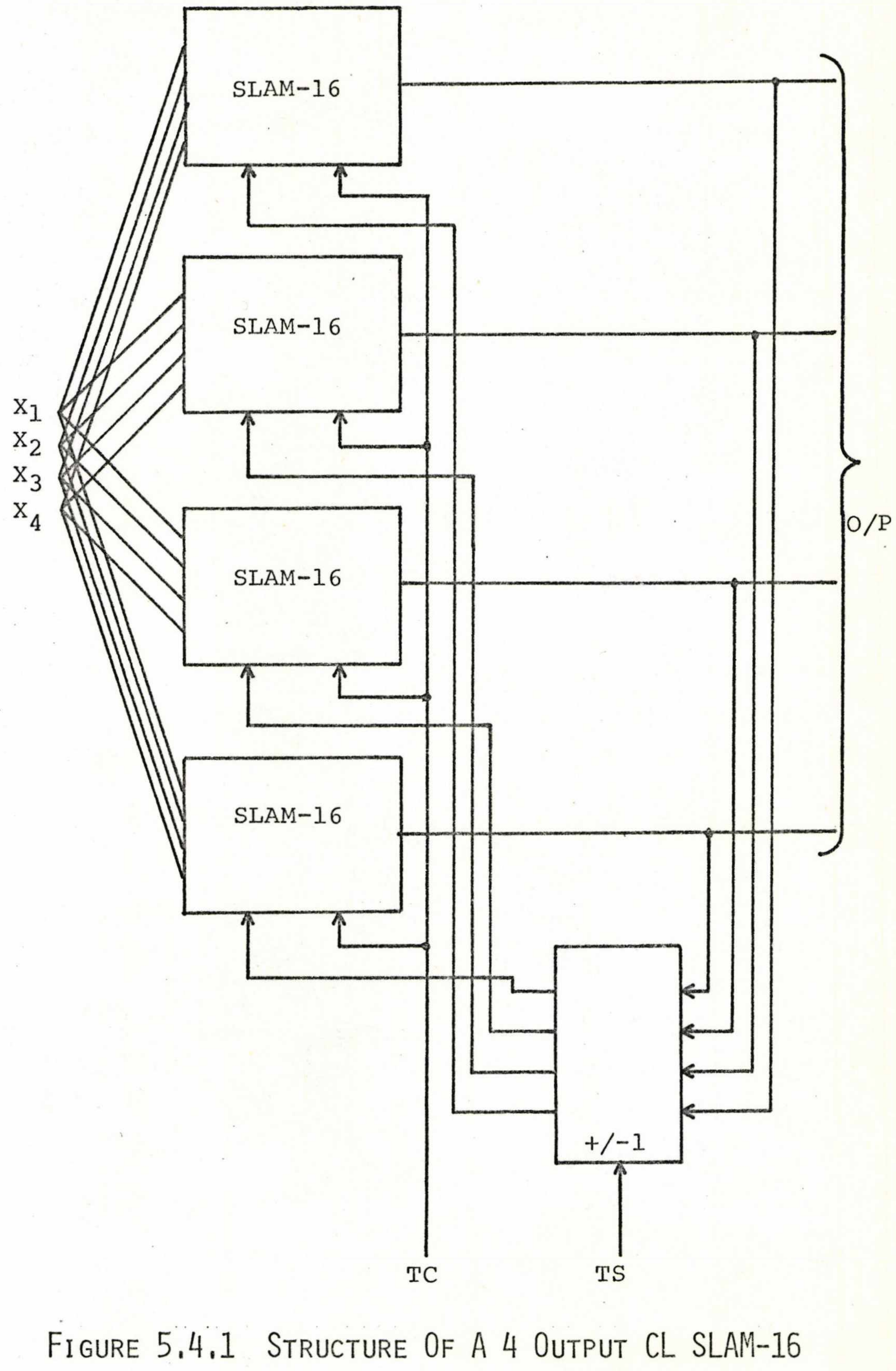


FIGURE 5.4.1 STRUCTURE OF A 4 OUTPUT CL SLAM-16

will output 14. The system similarly saturates at 0 when descending.

The concept of frequency of occurrence counting for n-tuple samples have been investigated by Bledsoe and Bisson(6) where they use computer stored matrices for recording the exact frequency of occurrence of each n-tuple input pattern for each class during training. The CL SLAM described above if not allowed to saturate is capable of a similar function.

The CL SLAM may be realised in hardware by two standard integrated circuits, a 64-bit memory and a 4-bit adder and it could easily be made on a single MSI integrated circuit. A patent has been applied for(39) for the concept of the CL SLAM.

The CL SLAMs that have been used for this project are 4 output CL SLAM-16s hence, they require 4 times the amount of store used by a SLAM-16.

5.4.2 CL SLAMs in a 2 class learning network.

The CL SLAM can be organised to perform the two class learning network problem described in section 5.1.2. The output numbers from the CL SLAMs are numerically summed to give an overall response. Teach sense and teach clock terminals are connected together, as before. If a particular 4-tuple pattern occurs for one class only, then this CL SLAM will eventually saturate and give a large output denoting an important feature. If a 4-tuple is common to both classes then the number associated with it will be incremented for one class and deincremented for the other and should remain near the neutral value '8' which indicates neither class.

With this learning network, the last pattern taught can only produce a maximum of 1/16th change in the overall responses, and also every bit (as opposed to the fraction in the last section) of all the input patterns is considered during teaching. The cost for this is that with 4 output CL SLAMs 4 times the amount of store is required. The performance of this learning network is investigated in section 5.6.

5.5 THE TERNARY SLAM ELEMENT.

5.5.1 The T.R. SLAM.

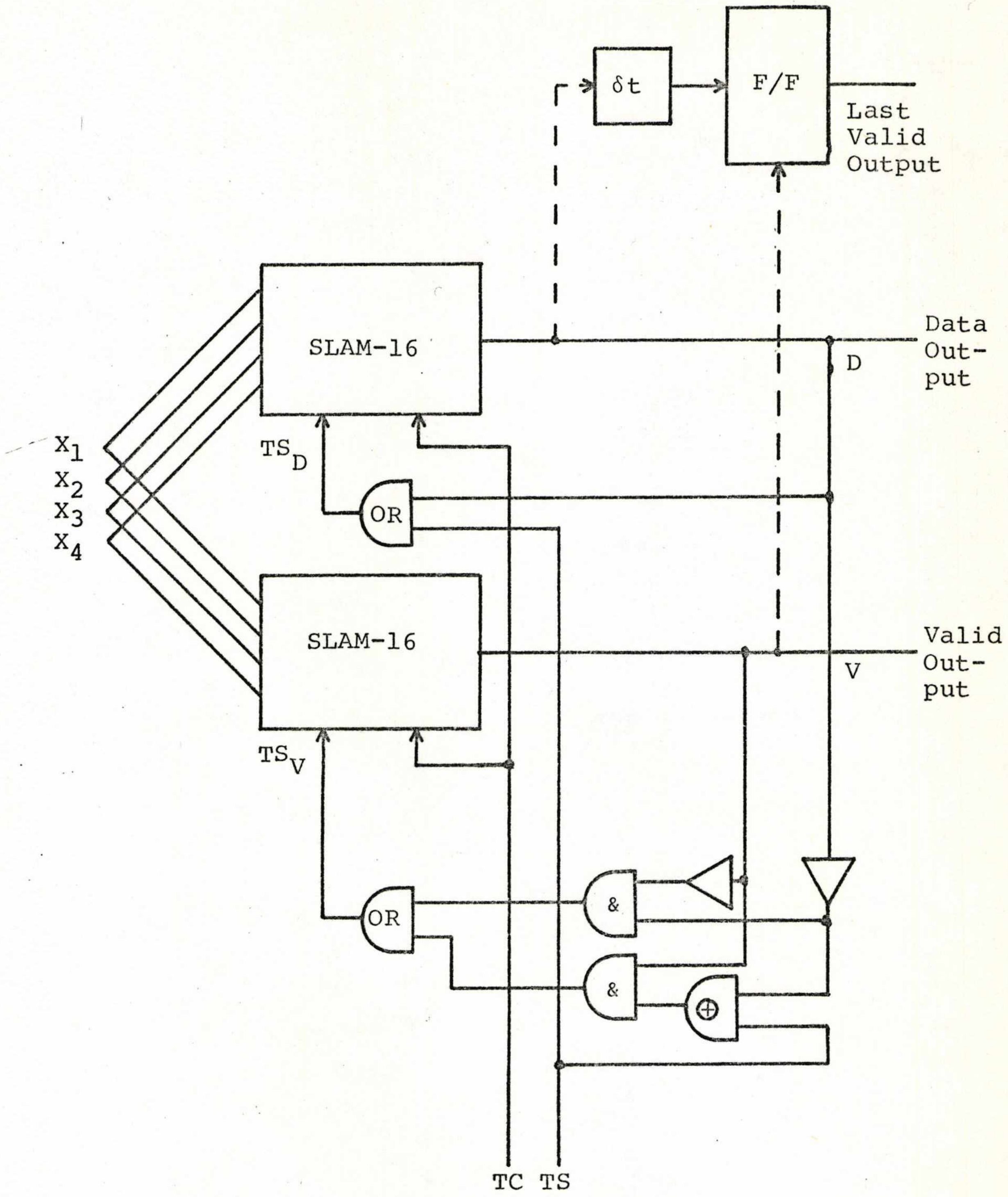
The Ternary or T.R. SLAM has been developed primarily for extracting features, especially where cycling is concerned, and the output of these SLAMs are used for feedback, as seen in section 4.2.3, for the classifier code net. The aim of the system is to distinguish between unseen, ambiguous and unambiguous n-tuple patterns. Only the latter are said to be 'valid'. There are two reasons for which the output of a SLAM could be considered as not being valid. Firstly, if the input pattern had not occurred during training, then the output is the initial value before training and hence of no informational value. This is the 'unseen' case. Secondly, the input pattern may be common to both classes and takes the value of the last class taught. This is the ambiguous case.

With a normal SLAM output there is no information indicating if an output is valid or not. The Ternary element has three output states 'valid' 1, 'valid' 0 and 'not valid'. (In practice, there are two distinct 'not valid' states but this fact is only used internally by the SLAM).

A T.R. SLAM-16 is shown in Fig. 5.5.1. Two SLAM-16s are used with their inputs coupled together, hence there are two bits of information associated with every input pattern. There are two binary outputs from the T.R. SLAM. One is called the 'data' output and the other the 'valid' output. There are four possible output states and these have been defined as follows.

Valid	Data			
0	0	=>	unseen	
1	0	=>	taught 0	unambiguous
1	1	=>	taught 1	
0	1	=>	overtaught (ambiguous)	

Initially, both SLAMs are reset and all outputs are in the untaught state. When taught a 0 or a 1 the valid output becomes true and the data output indicates the value taught. If the output is valid and it is taught to output the opposite value to that which it is already



$TS_D = TS + D$

$TS_V = \overline{V}.\overline{D} + V.(T \oplus \overline{D})$

FIGURE 5.5.1 ORGANISATION OF A T.R. SLAM-16

valid for the fourth state, the overtaught state, is entered. This last state is permanently 'not valid' and no further teaching will have any effect until the T.R. SLAM is reset again.

The above output states can be realised in hardware by the logic shown in Fig. 5.5.1. The two teach terminals are connected together and form a common teach clock terminal. The extra logic generates the required values for the two teach sense terminals and the Boolean equations which are realised by this logic are also given in Fig. 5.5.1.

To construct T.R. SLAMs the teach sense terminals must be accessed separately and this is not possible with the present form of Minerva, hence, all experiments with with T.R. SLAMs have been done by computer simulation.

5.5.2 T.R. SLAMs used in the classifier.

When using T.R. SLAMs in the code net for the classifier (see section 4.2.3), a lot of overteaching was experienced, hence for the majority of the time which the input sequence was applied, the output of each T.R. SLAM was not valid. However, the output of the SLAMs is fed back to the classifier and this feedback must have a value even when the T.R. SLAM output is not valid. To overcome this problem the 'last valid output' is remembered and is fed back to the classifier until another valid output occurs.

This additional memory may be added to the T.R. SLAM element by the use of a D flip-flop as shown by the broken lines in Fig. 5.5.1. When the valid output is true then the output of the flip-flop follows the input (which is the data output). When the valid output is false then the last valid data value is retained by the flip-flop.

5.5.3 Two features of T.R. SLAM learning networks.

When using a collection of T.R. SLAMs for a learning network two additional features of the learning network can be observed. Firstly, after training, the stores of the SLAM elements can be examined and the number of untaught, valid taught and overtaught states can be obtained. From this information some measure of the usefulness of individual n-tuples can be obtained. Also, some measure of the expected performance of the learning network can be obtained from this information. This feature has been used in analysing the classifier (see section 4.3.3). Secondly, when, after training, the learning network is tested the number of valid outputs which occur can be used to generate a confidence level for the response of the learning network. This feature is used by the classifier and is also considered in the experiments in section 5.6.

The main disadvantage with the T.R. SLAM is that if it is taught a rogue pattern (e.g., a pattern of the wrong class) then many of the useful valid states will be overtaught and made permanently invalid.

If it was realised that a rogue pattern had been taught some recovery could be made by setting all the overtaught non-valid states to untaught non-valid states and then training could be continued. In practice, this could easily be achieved by scanning through the 16 states of each T.R. SLAM and changing the overtaught ones as they occur.

5.5.4 T.R. SLAMs in a 2-class learning network.

The T.R. SLAM can be organised as a two-class learning network (i.e., for the task described in section 5.1.2). The teach-sense terminals and teach-clock terminals are connected together as with the normal SLAM learning networks. The output response must be calculated in a different way. For a normal SLAM network the number of outputs which are 1 are summed and expressed as a fraction of the total number of SLAMs. For a net of T.R. SLAMs the number of valid outputs which are 'one' must be expressed as a fraction of the total number of valid outputs. The total number of valid outputs should also be noted to give a confidence figure for the response.

For example, if for a learning network of 64 T.R. SLAMs only one output was valid, then the response would be 100% for the class of that output. This decision has been made by considering only one unique 4-tuple of a

256-bit input and it is important to indicate this fact. Noting the number of valid outputs will achieve this e.g., if the output is stated to be 100% for a class when only 1/16th of the learning net output is valid.

For a learning network of N TR SLAMs used in a two-class classifier it is possible to express both the data response and the valid response in terms of a total response 'R' which may be defined by the following formula

$$R = \left(D + \frac{N-V}{2}\right) \frac{100}{N}$$

where R is the total response (% of N)

N is the total number of T.R. SLAM elements

D is the number of 1 valid outputs

V is the total number of valid outputs

(e.g., if there is only 1 valid output and its value is 1 and N=64, then the total response R=50.9%).

The above learning network has not been investigated directly in the tracking system experiments. However, if one considers the maximum-response learning system described in section 3.3 the structure used there is very similar to a net of T.R. SLAMs. For the case where the input mappings are identical (see section 3.3.5), the performance of that tracking system is identical to that of a learning network of T.R. SLAM-16s when a <u>two-level threshold</u> is set at the output.

5.6 EXPERIMENTS WITH 2-CLASS LEARNING NETWORKS.

5.6.1 The structure of the experiments.

To illustrate the properties of the different SLAM elements described in this chapter and to compare the performance of these elements when used in the two-class learning mode two experiments have been conducted. The learning networks which have been used are described in the following sections

Normal SLAMs	5.1.2
Probability Learning Networks	5.3
CL SLAMs	5.4.2
T.R. SLAMs	5.5.4

These experiments were conducted with the subsystem shown in Fig. 5.6.1. More details of this subsystem are given in section 6.4.2.

When a 16x16 bit input pattern is presented to the subsystem it is randomly mapped into a second 16x16 pattern. This second pattern is then presented to the different learning networks. Hence, the same 4-tuples are sampled by all the learning networks. The teach-clock and the teach-sense terminals of the different learning networks are commoned together. Hence, all the learning networks receive the same input data and the same teaching stimulus. All the outputs are summed to form a response as described in the relevant sections above.

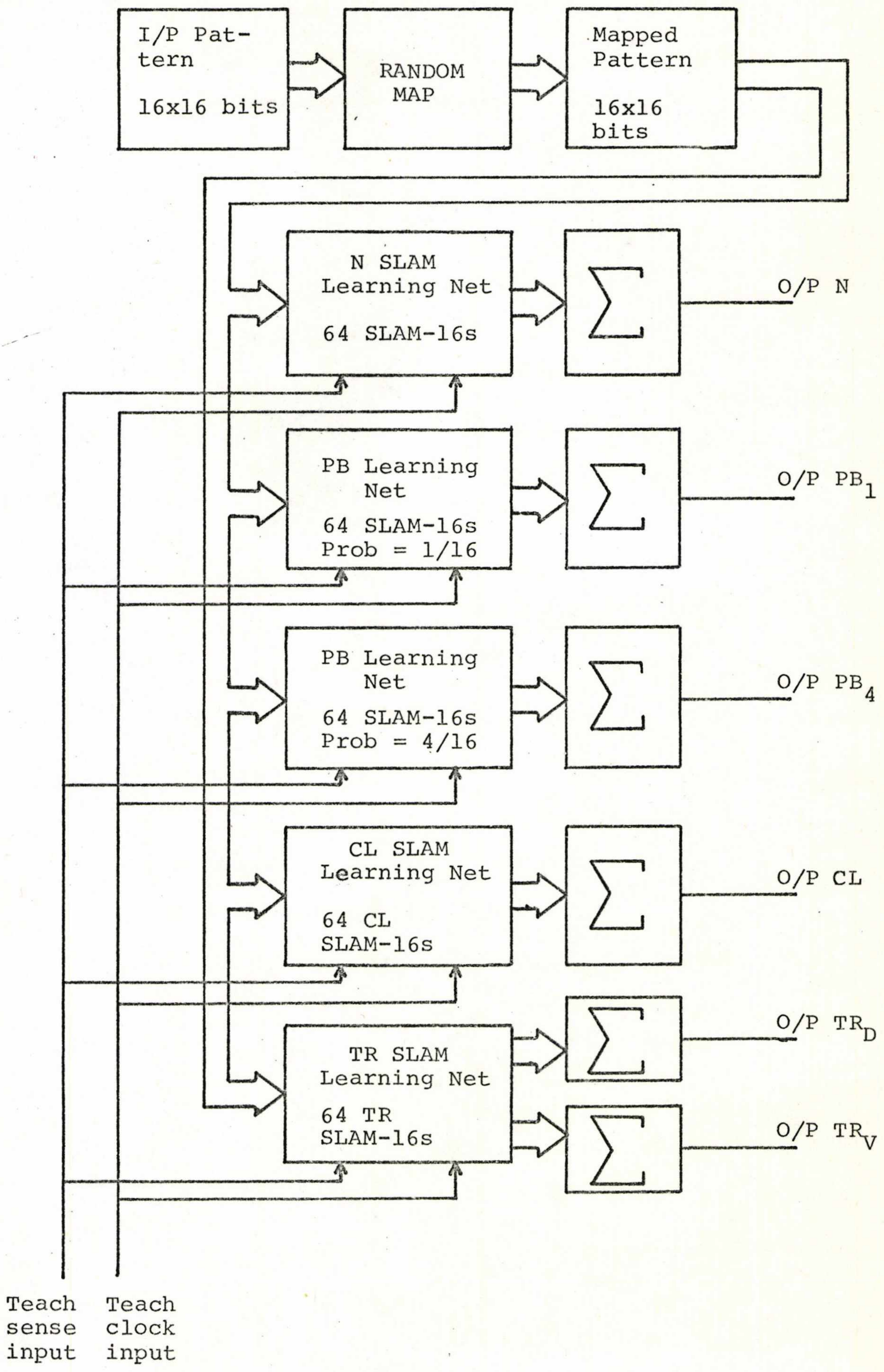

I/P Pattern
16x16 bits
RANDOM MAP
Mapped Pattern
16x16 bits
N SLAM Learning Net
64 SLAM-16s
O/P N
PB Learning Net
64 SLAM-16s
Prob = 1/16
O/P PB$_1$
PB Learning Net
64 SLAM-16s
Prob = 4/16
O/P PB$_4$
CL SLAM Learning Net
64 CL SLAM-16s
O/P CL
TR SLAM Learning Net
64 TR SLAM-16s
O/P TR$_D$
O/P TR$_V$
Teach sense input
Teach clock input

FIGURE 5.6.1

Before an experiment is conducted, the SLAM stores are initialised in the following way. For the learning networks which contain normal SLAM-16s half of the SLAMs are set and half are reset hence a 50% output response will be obtained for any input pattern. The CL SLAM learning network has all its stores set at 8 hence this learning network will also output a 50% response to any input pattern. The T.R. SLAM learning network has all its SLAMs reset hence no valid outputs which implies a 50% response.

5.6.2 Dependence on last pattern taught.

The first experiment is designed to demonstrate how the last pattern taught affects the performance of the learning networks.

The input data is in the form of two sets of handwritten characters on 16x16 matrices. The characters, chosen for the two-class, are '3's and '8's which is a difficult task as there is only a small Hamming distance between them. Each class contains 260 patterns.

The experiment is conducted in the following way:

1. The first 3 is taught to the learning networks to output a high response.
2. The average response of the learning networks for the next ten (i.e., untaught) 3s is obtained.

3. The average response for the first ten 8s is obtained.
4. The first 8 is taught to output a low response.
5. The average response for the next ten 3s is obtained.
6. The average response for the next ten 8s is obtained.
7. The second 3 is taught for a high response.

The above sequence is repeated 250 times, then the learning networks will have been taught 250 patterns for each class.

The average response to the next ten untaught patterns is used to measure the performance of the learning networks with respect to that class. A standard test set was not used as with only ten patterns it may not be typical of the class.

From the results of this experiment four graphs have been drawn for each learning network by plotting the average response versus the number of patterns taught.

The graphs show the following response:

LALH	(Low After Last High) this is the performance of the learning network to the low response class (8s) after the last pattern taught was of the high response class (3s).
HALH	High performance after last class taught is high.

LALL	Low performance after last class taught is low.
HALL	High performance after the last class taught is low.

These graphs are shown in Figs. 5.6.2 to 5.6.7 and we recall that the object is to separate the high from the low.

In Fig. 5.6.2 the graphs from the normal SLAM learning network are shown. The last pattern taught alters the response to both classes by about 20%. This implies that about 20% of the input 4-tuples are common to both classes. There is a lot of confusion between LALH and HALL graphs in the range 40-60% response. This means that a threshold cannot be set at 50% to separate the two classes. It may be possible for a threshold to be set at another value but to do this one must know which class was the last one taught. After the first ten pairs of patterns have been taught, further teaching has very little effect on the performance.

In Fig. 5.6.3 the graphs for the probability learning network with n=1 are shown. For this case, learning is much slower as only 1/16th of the learning network is taught at a time. About 100 pairs of patterns are required to teach the learning network so that further teaching has little effect. The shift due to the last pattern taught is in the order of 1-2% for the responses from both classes. This agrees with the expected value

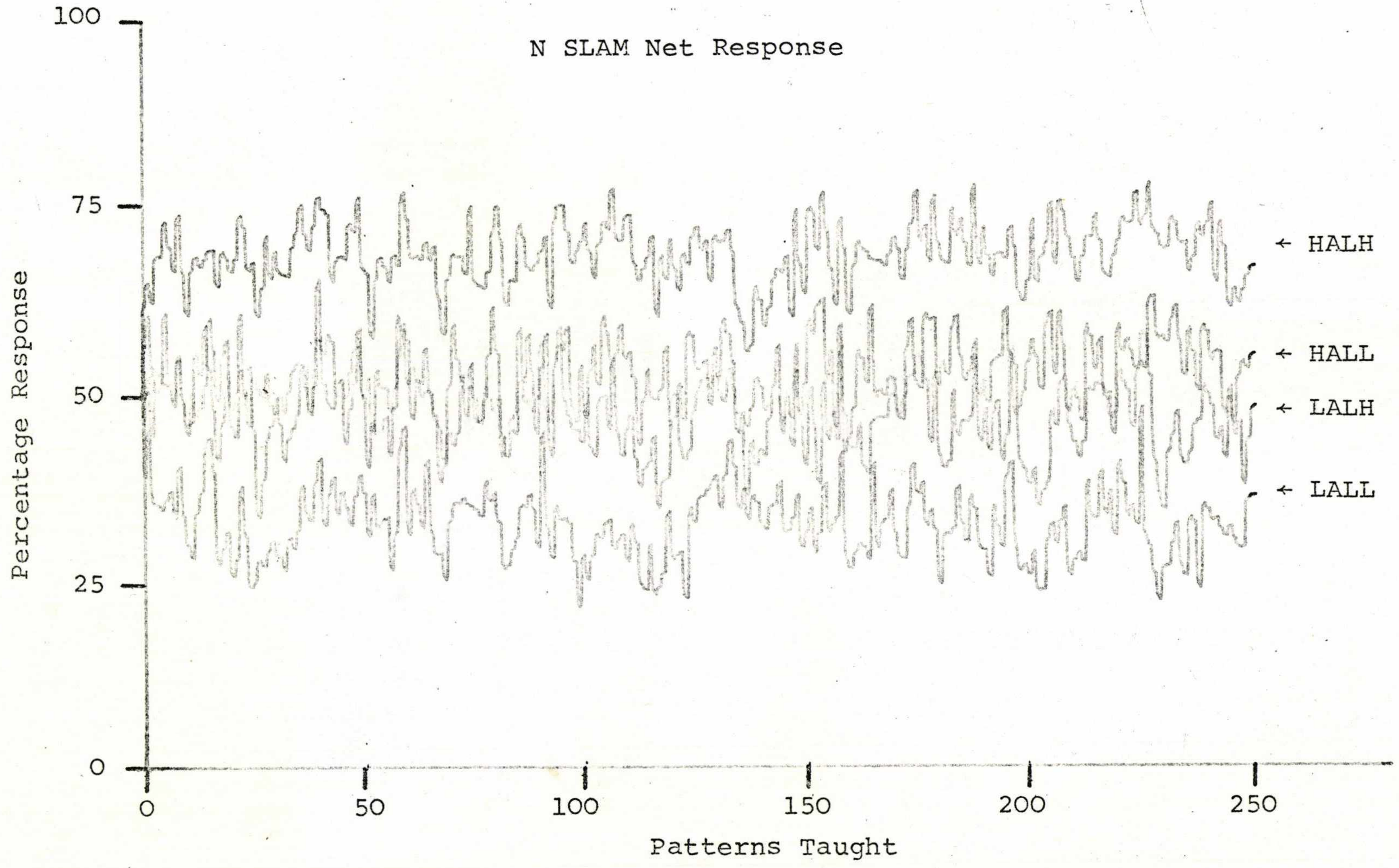
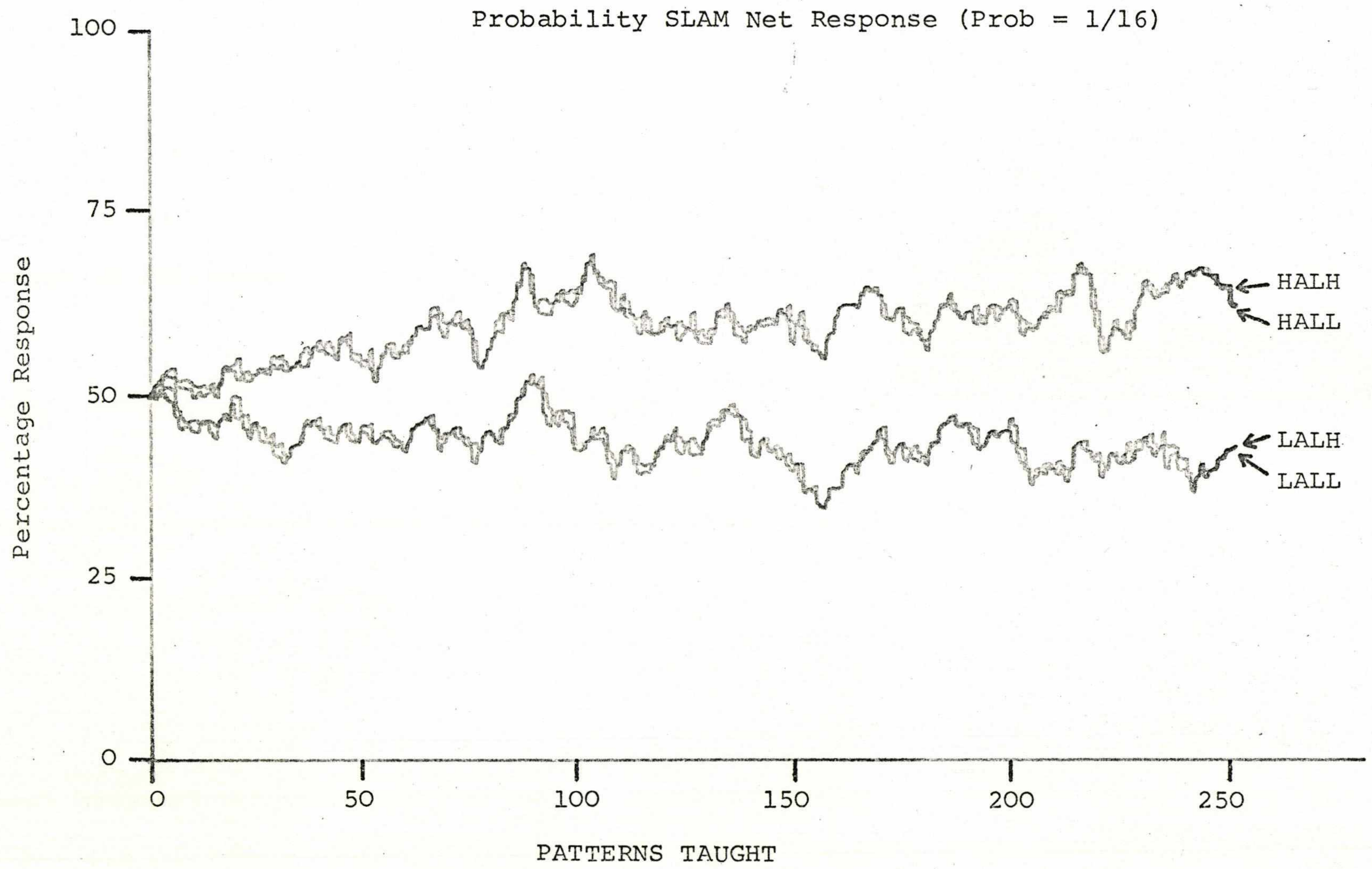
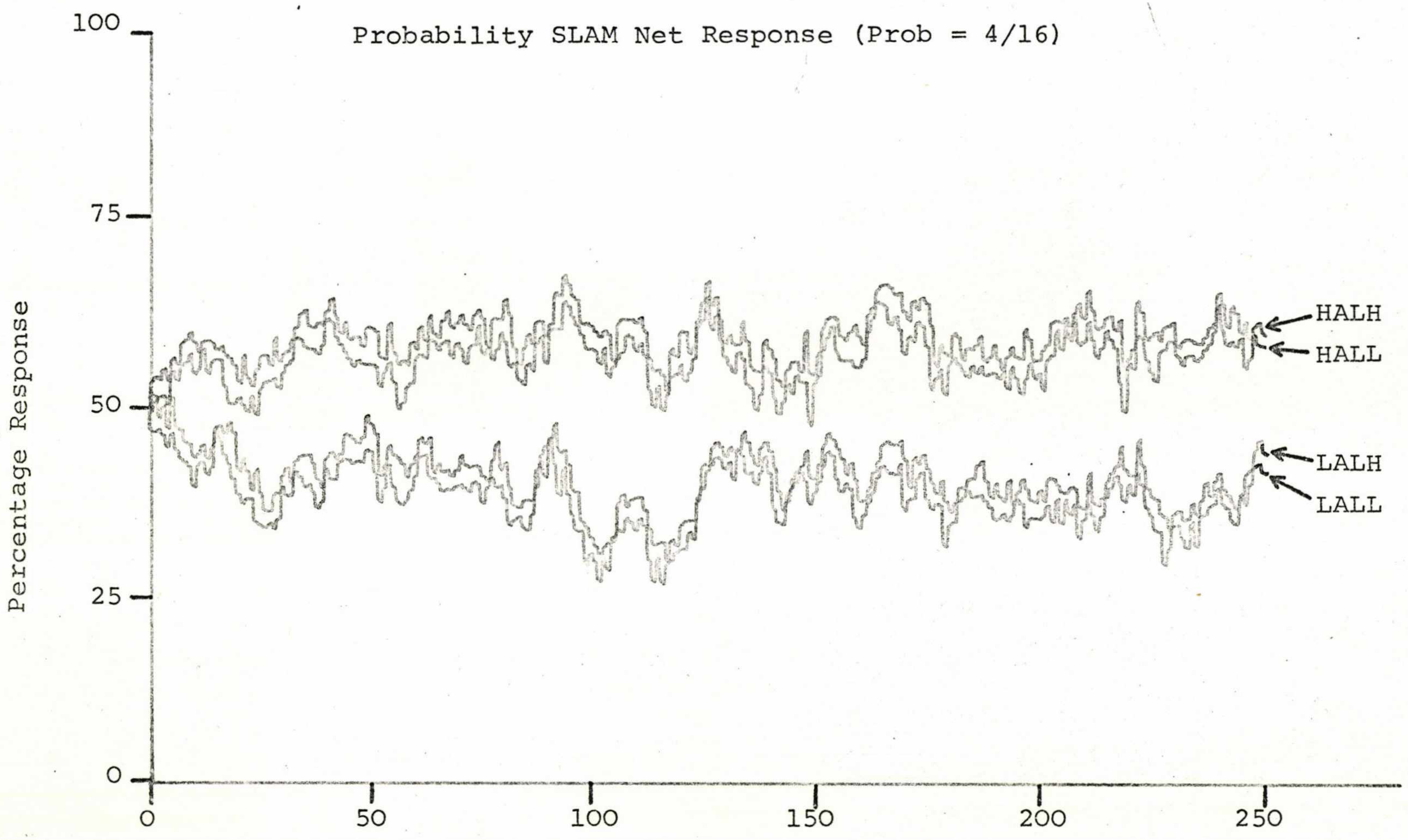

N SLAM Net Response
100
75
50
25
0
Percentage Response
0
50
100
150
200
250
Patterns Taught
← HALH
← HALL
← LALH
← LALL

FIGURE 5.6.2



Probability SLAM Net Response (Prob = 1/16)
100
75
50
25
0
Percentage Response
0
50
100
150
200
250
PATTERNS TAUGHT
HALH
HALL
LALH
LALL

FIGURE 5.6.3



PATTERNS TAUGHT

FIGURE 5.6.4

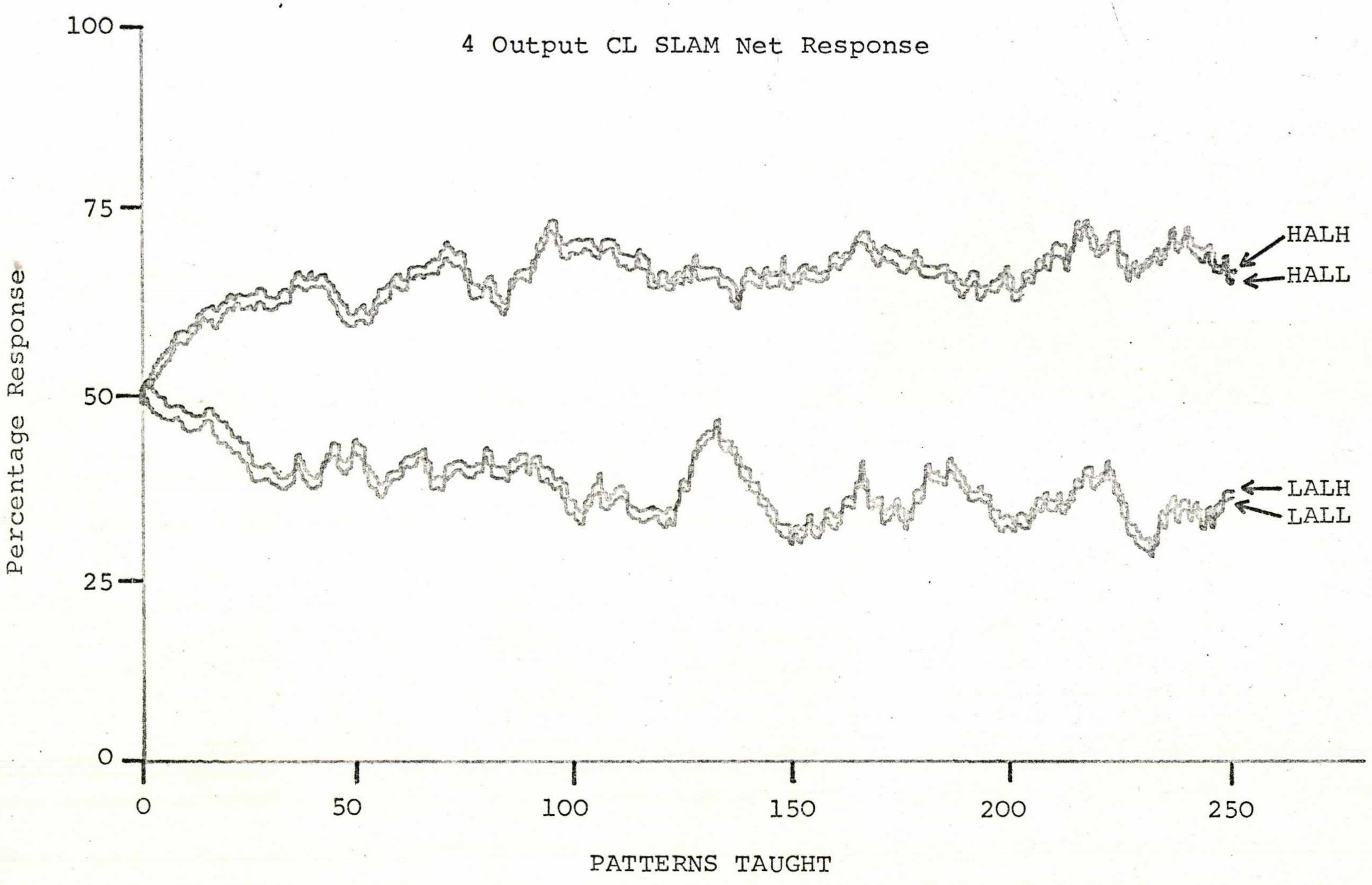

4 Output CL SLAM Net Response
Percentage Response
100
75
50
25
0
0
50
100
150
200
250
PATTERNS TAUGHT
HALH
HALL
LALH
LALL

FIGURE 5.6.5

16 Output CL SLAM Net Responses

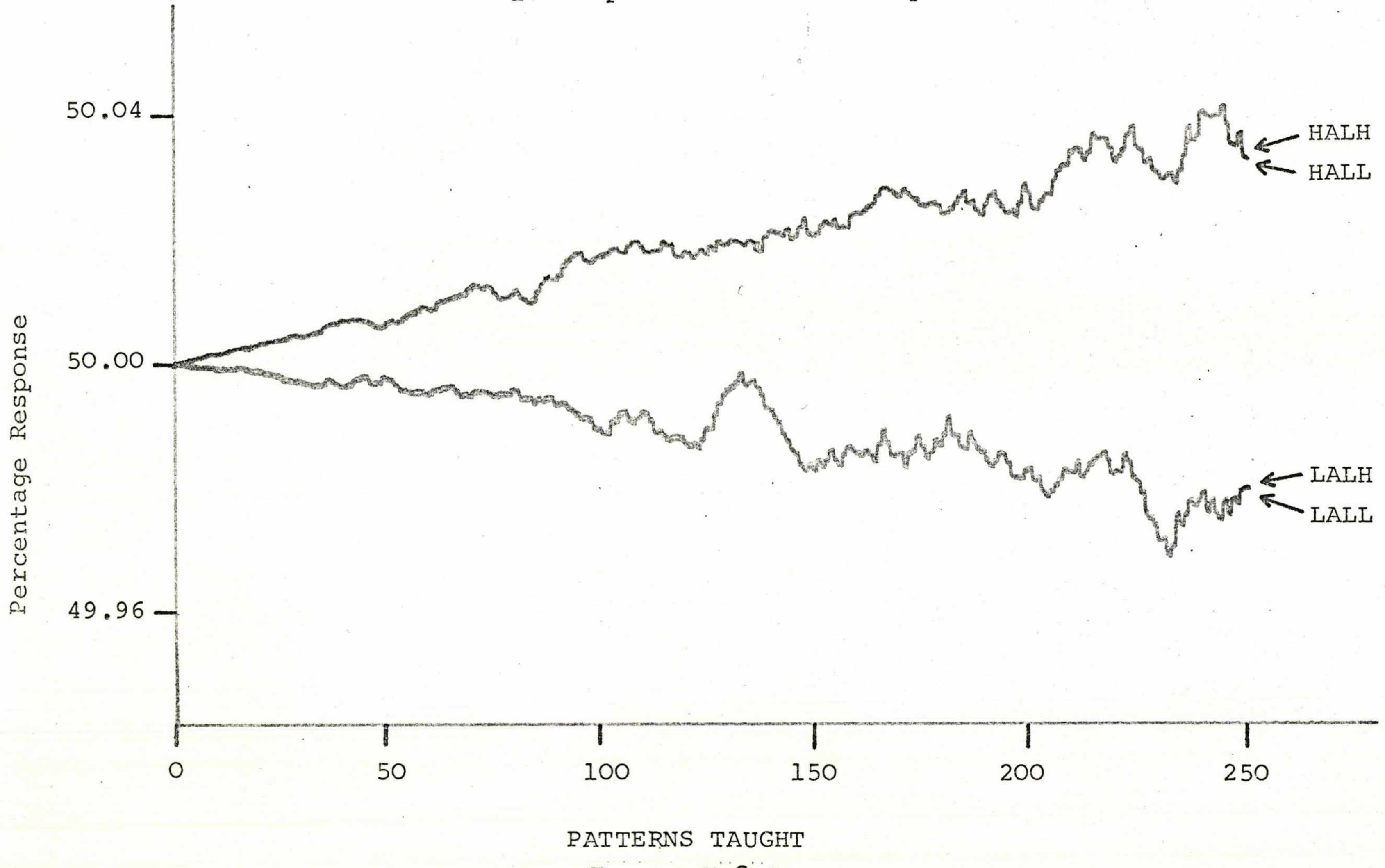


FIGURE 5.6.6

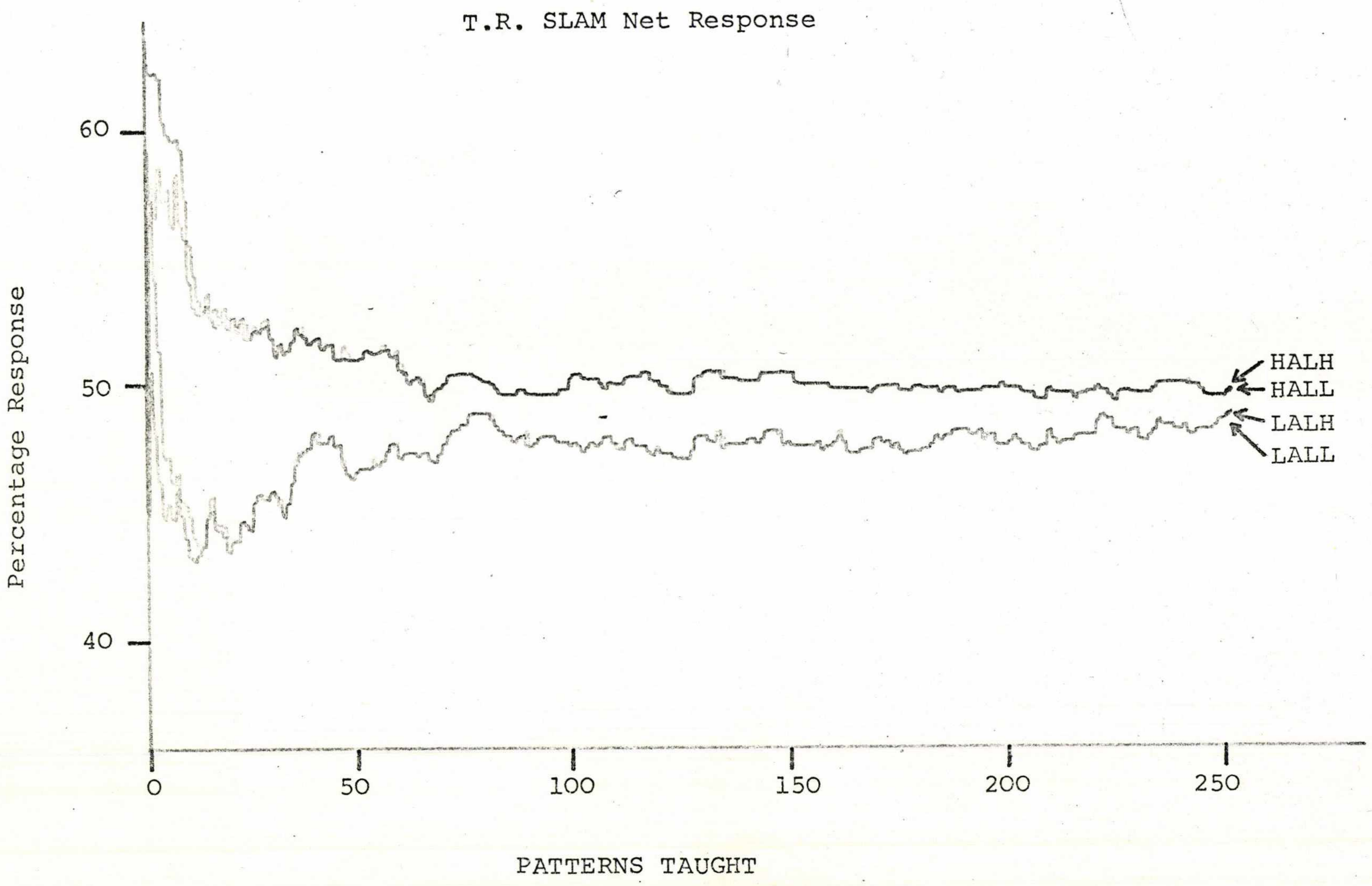

T.R. SLAM Net Response
Percentage Response
60
50
40
0
50
100
150
200
250
PATTERNS TAUGHT
HALH
HALL
LALH
LALL

FIGURE 5.6.7

which is 1/16th of the value for the normal SLAM learning network i.e., 1.25%. The graphs for the probability learning network with n=4 are shown in Fig. 5.6.4. The separation between the classes is about the same. However, only about 30 pairs of patterns need be taught before further teaching has little effect. The shift due to the last pattern taught is about 5% which is the value expected i.e., ¼ of the effect which was observed with the normal SLAM learning network.

In Fig. 5.6.5 the graphs for the CL learning network are shown. In this case there is a rapid improvement in the performance for each class while the first 30 pairs of patterns are taught. Then the CL SLAMs begin to saturate and further improvement is much slower (~5% for each class after the next 200 pairs of patterns have been taught). There is about 2% change in performance due to the last class taught which agrees with the expected value of 1/16th of the normal SLAM learning network response i.e., 1.25%. The separation between the classes is better for the CL SLAM learning network than for the other learning networks tested.

To find the effect of not letting the CL SLAMs saturate,this experiment was performed on a learning network of 16 output CL SLAM-16s. These CL SLAMs can be incremented 32,768 times in either direction before saturation occurs. Hence, with a training set of 250 patterns of each class saturation could not occur. The graphs for this learning network are shown in Fig. 5.6.6, the Y axis scale has been magnified as the maximum change in response possible is only 250/32,768=0.8%. The effect of the last class taught is negligible. There is a steady improvement in the performance of this learning network while the 250 pairs of patterns have been taught. This learning network uses 16 times the amount of active store which is used by the normal SLAM learning network.

In Fig. 5.6.7 the graphs for the T.R. SLAM learning network are shown. The Y axis scale has been magnified for clarity. It must be remembered that the responses are obtained in a different way for this learning network than from the others in that the concept of valid outputs is implemented. After the first 20 patterns have been taught the shift due to the last pattern taught becomes negligible. The separation with this system becomes worse as teaching progresses. This happens because the learning network cannot forget anything it has been taught. In this experiment the input data is very diverse in Hamming distance and contains many poor examples of both classes. This 'saturation' of the learning network could be partially overcome by employing an algorithm which would reset all the overtaught states (as described in section 5.5.3) after a set number of patterns have been taught.

5.6.3 Classification ability.

The second experiment was designed to demonstrate and compare the ability of the learning networks to classify the characters of each class.

For this experiment the learning networks were organised to have equal amounts of active store. The CL learning network contained 64 4-layer SLAM-16s, the T.R. SLAM network contained 128 T.R. SLAM-16s and the probability and normal SLAM learning networks contained 256 SLAM-16s each (hence, they cover the input matrix four times). By increasing the active store in this way one would not expect an improvement in the performance of the system, one would merely hope that the results obtained with the system would be more consistent.

The learning networks were taught 50 patterns of each class alternately (i.e., in the same way as for the first experiment). The number of errors made with the training set of 100 patterns and with a test set of 100 patterns (50 of each class) was then obtained.

These results are as follows:

	Learning Network			
	Normal	Probability n=4	T.R.	CL
Errors for training set of 100 patterns	31	14	0+2	2
Errors for test set of 100 patterns	46	32	13+5	10

Two figures are given for the T.R. SLAM learning network. The first is the number of wrong classifications and the second is the number of occasions when there were no valid outputs hence, no classification could be made. It is not possible for the T.R. SLAM learning network to misclassify any of the training set.

For the normal SLAM learning network all the errors were made for the first class i.e., the opposite class to the last class taught.

For the first experiment the average performance of the probability SLAM learning network appeared similar to the performance of the CL SLAM learning network (see Fig. 5.6.4 and Fig. 5.6.6). However, the actual performance of the CL SLAM network is in fact far superior (10 errors for the test set compared with 32 for the probability learning network).

The input data used was of a very poor quality to ensure that a comparison between the different learning networks could be made. Examples of these patterns are given in Fig. 5.6.8.

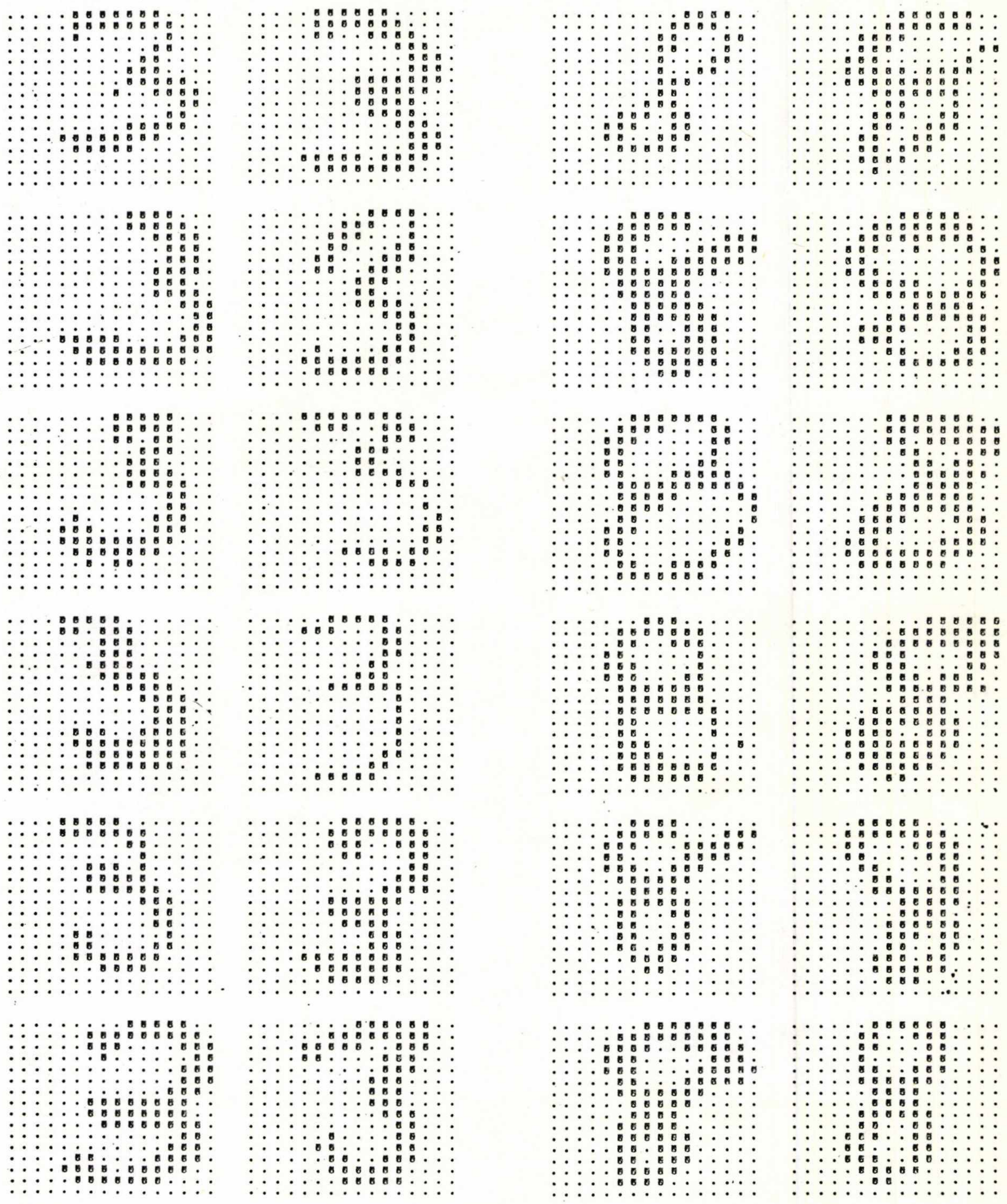


FIGURE 5.6.8

CHAPTER 6
THE SOFTWARE SYSTEM

In this chapter the software used for the experiments in this project is considered in detail. The aims and basic structure of the software system have been outlined in section 2.3.

In the first part of this chapter the main program and executive modules are described in detail. In the second part of this chapter the structure and function of the main experiment subsystems is described and a particular subsystem (experiment 9 which is used in Chapter 5) is considered in detail. A list of the routines which are available to a user when writing an experiment subsystem is given in Appendix 2.

6.1 DETAILS OF THE SOFTWARE SYSTEM.

6.1.1 Structure of the system.

The structure of the system and the design considerations which led to this structure are considered in section 2.3. The final structure consists of a set of modules which may be classified in the following way:

1. A main program, this sets up the workspace and allows on-line access and manipulation of pattern data within this workspace.

2. An executive program which controls all the peripherals.

3. An experiment subsystem to conduct the experiment.

4. A library of subroutines to be used by the above three modules.

5. An optional debugging module which may be loaded into the workspace when needed.

The main program and the executive program are described in sections 6.2 and 6.3 and then experiment sub-systems are described in section 6.4. The library is a set of routines called by the other modules. When operations are carried out by library routines this is mentioned in the descriptions of the modules. The Debug program is described elsewhere (3). A special version, developed for the system, contains all of the functions of the normal version, but is linked to the system via the executive module.

Examples of some of the functions available are given below:-

1. Break points - Programs may be stopped, examined and restarted at locations defined by the user.

2. Core Insertion & dumping - DAP-like programs or data can be typed directly into the store, also the store can be dumped in either data or instruction formats.

3. Core Searching - Selected areas of core store can be checked for particular instructions or data words (or parts of words) and the locations where these occur will be output.

6.1.2 Inputting information to the system.

The system has been designed so that when a user wishes to input information to the system via the command device there are three standard formats which he can use.

1. A one- or two- character mnemonic. This format is usually used in a command status to specify the different functions available.

2. An integer number. This is assumed to be decimal but by preceding it with 'B' it will be binary or ''' it will be octal. After the number has been input it must be terminated by a space or ',' or '.' .

3. Yes/No answer. This format is usually used when a binary decision is required by the system from the user. The system outputs a question which the user answers by inputting 'Y' or 'N'.

Everywhere, except in the experiment subsystem, only the above three formats have been used. Functions have been specified by two-character mnemonics to reduce the typing for the user and to simplify decoding in the program. Except for the most frequently used functions, the system outputs a message after a mnemonic function has been input which states either the action taken or what further information is required.

Binary pattern information may also be input by the system. The format of this information depends on the peripheral concerned, there are two standard formats for representing a 16x16 bit pattern.

1. Paper tape format. This consists of a 'rubout' (i.e., all 8 holes punched) followed by 32 8-bit frames to specify the pattern.

2. Magnetic tape format. This consists of 16 16-bit words to specify the pattern.

6.1.3 Computer organisation.

A summary of the main peripherals used by the system is given below. These peripherals will be referred to in later sections.

1. Teletype. A standard ASR 33/35 teletype: 10 characters/ second. Information may input or output in either character or pattern (paper tape) format.

2. Vista. A 'CASE Vista' visual display terminal which has a solid state keyboard and a cathode ray tube display of 20 lines of characters. Character information may be input from or output to this terminal. This will be referred to as the Vista in the rest of the chapter.

3. Paper tape stations. There are two paper tape stations which can read and punch paper tape. The paper tape may be in character format or pattern format.

4. Line Printer. The line printer has a 96 character line and can output 300 lines a minute.

5. Magnetic tape unit. A standard Honeywell 50,000 bits/sec. magnetic tape handler. It treats data in core transfers of 16-bit words and requires a software handler.

The above devices are standard computer peripherals. There are some non-standard peripherals which are listed below.

6. T.V. Camera. This requires a software handler to obtain a 16x16 matrix from a scene. The data is stored as a pattern in the computer. Further details are given in section 2.2.

7. Minerva. A hardware learning machine closely linked to the computer. It always completes an operation within 1 computer cycle hence, it is not programmed as a peripheral. It's instructions are treated as an extension to the standard instructions of the computer. Details of the organisation of Minerva are given in section 5.2. Hardware details are described elsewhere(17).

8. V.D.U. A Visual Display Unit is available which displays a 16x16 binary pattern of information on an oscilloscope screen. This device time shares whatever program is being executed and continuously displays 16 consecutive words of store. The time sharing has an overall effect of reducing the effective speed of the computer by 5%. Hardware details of this device are given elsewhere(38).

9. Analog Plotter. There is an interface which contains 2 10-bit D to A convertors. This may be connected to a Hewlett Packard analog plotter. This has been used for plotting learning network responses and for plotting scan paths for the tracking system.

10. Disc. There is a disc unit which has a non-standard handler. This has not been completed yet and is not available to computer users unless it is accessed by an operating system 'ADMOS' which has recently been developed at this University(4). This may be accessed by the special executive described in section 6.2.5.

6.2 THE EXECUTIVE PROGRAM.

6.2.1 Structure of the executive.

The executive program is 1,142 (Octal 2,200) locations long. The executive contains all the routines for controlling the peripherals. It has a command status which is entered when a program break occurs.

A program break may be caused by one of three occurrences

1. if octal 221 (character Q with control on the teletype) is input on the command device. This is the usual way to enter the command status;
2. if the start button on the computer console is pressed;
3. if an interrupt occurs from a device not enabled by the system. (This would usually indicate that there is a hardware fault).

The peripherals may be divided into two distinct types: the standard peripherals which use 8-bit words and the special peripherals which require software handlers.

6.2.2 Standard peripherals.

The peripherals relevant to this section and their mnemonics which are used in the system are as follows:

	INPUT	OUTPUT
Teletype	TT	TT
Vista	VI	VI
Paper tape Station 1	PR	PP
Paper tape Station 2	R2	P2
Line printer	-	LP
(Null)	-	NO

Information may be input to and output from the system by two channels; a command channel and a data channel. The command channel is intended for inputting commands and for outputting messages from the system. The data channel is intended for inputting and outputting pattern data. There are two additional output routines, one for the line printer and one for the Vista. These allow fast dumping of data. There is a special output channel which is labelled the mimic channel, this monitors both the input and the output information which passes through to command channel. This channel is primarily intended for creating a hard copy via the line printer or paper tape punch when the command device is the Vista so that a permanent record of the experiment may be obtained.

A channel has the property that it may be assigned to any one of the peripherals listed at the beginning of this section. Initially, when the system is started, the channels have the following default devices.

	INPUT	OUTPUT
Command Channel	TT	TT
Data Channel	PR	PP
Mimic Channel	-	NO

Hence, commands are input and messages are output on the teletype and pattern data is input and output on paper tape station 1. Data may also be output to the Vista or to the line printer by their special routines.

When a user wishes to write a subsystem two ways of inputting information are available to him (command channel and data channel input) and four ways of outputting information (command channel, data channel, line printer output and Vista output). To input data there are two routines, one for each channel. When a character (an 8-bit word) is requested, the appropriate routine is called. When the routine has obtained a character from the peripheral it returns to the program with the character in the right half of the 'A register', to obtain the next character the routine must be called again. Outputting, conducted by one of four routines, is achieved in a similar way. The character to be output is loaded into the right half of the 'A register' and the routine is called, a return to the program occurs when the character has been output to the peripheral.

Hence the executive program deals only with single characters at a time. The library contains many routines which use the above mentioned routines and enable the user to input and output information at a higher level. For example, for the command channel there are routines for inputting mnemonic commands and checking these with function lists, for inputting and outputting numbers, for outputting messages etc. For further details see Appendix 2.

With most operating systems these peripherals are time shared in an interleaved manner. However, for this system they have not been time shared for the following reasons:-

1. The time saved by time sharing would be very small for most of the processor time is usually spent in internal processing and very little time is used for dumping results while this processing is in progress.

2. The executive program would have to be much more complex and would also require much more store.

3. A useful feature of the system is that a user may stop the program when it is running and slowly step through it to see exactly what is happening. This is not possible if time sharing is allowed.

A user may change the allocations of the channels to the peripherals from the program by calling routines or on-line when in the executive command status.

6.2.3 Command status functions.

The executive command status allows channels to be assigned in the following way. One must first define the 'input' (or the 'output') to a peripheral then the channel may be assigned to the defined 'input' (or 'output') peripheral. For example, to assign the output of the command channel to the line printer one would type DO LP (define output to line printer), OD (set command output to defined output). The command output may be returned to the default device (i.e., the teletype) by typing ON (command output normal).

The mnemonics for assigning the different channels are given below

	INPUT		OUTPUT	
	DEFINED	DEFAULT	DEFINED	DEFAULT
Command Channel	ID	IN	OD	ON
Data Channel	RD	RN	PD	PN
Mimic Channel	-	-	MD	MN

There are several options which may be set or reset by the user in the executive command status and these are as follows:-

1. Cancel Messages (set by CM, reset by PM). When this option is set there is no output on the command channel. This does not effect the mimic channel.

2. Tape Control (set by TC, reset by RC). When this option is set the command input is taken from paper tape reader number one. This is conceptually different from assigning the command input to the paper tape reader. This appears different to the user in two main ways. Firstly, all commands input from the paper tape reader are mimicked on the command output device. Secondly, after a program break on the paper tape the system remains under tape control. If a user causes a program break then control returns to the defined command peripheral. This is a very important option, its use is demonstrated in the example in section 6.4.2 .

3. Vista Control (set by VC, reset by NC). When this option is set the Vista completely replaces the teletype. Hence any references made to the teletype will be interpreted to mean the Vista. The Vista is a more convenient command device than the teletype and one very rarely wants to use both these peripherals at the same time.

4. DEBUG Option (set by $D, reset by ND). $D should only be input when the debug program has been loaded. This command makes the links with the debug program and allows DEBUG commands to be input within the system.

5. Special Commands (set by SC, reset by NS). This option is usually set. It checks each character that is input from the command device for the appearance of special characters. These characters are listed below with the functions that they incur once they are detected.

& This character forces the system into the main program command status.

* This forces the system into the experiment sub-system command status.

$ This puts the system into the command status of DEBUG if it has been loaded.

@ This command must be followed by an argument n($0 \leqslant n \leqslant 7$). It allows 8 different starting locations for the VDU to be remembered. When @n is input the VDU is directed to the correct area of core store and neither command character is passed to the system. Hence the system is in exactly the same state as it was before the command was given. Sometimes a user may wish to input one of the above characters for a different function e.g. in typing a heading. To do this, the option must be reset.

6.2.4 Special peripherals.

The magnetic tape is really a standard peripheral however, it requires a lot of software handling. As it is rarely used an optional subsystem has been written to operate it. This subsystem is separate from the executive and it is linked to the main program when it is used.

The special peripherals which are handled within the executive are the television camera and the VDU. The graph plotter has not yet been written into the executive however, it only uses a few instructions and any experiment subsystem which uses the plotter can easily contain these instructions within it.

The television camera is controlled by one routine. This routine outputs all the parameters to the camera, obtains a matrix from the camera, stores it in a 16x16 buffer and returns. The camera may be operated in one of two modes:

1. The non-time sharing mode, in which the camera only inputs information from the scene when commanded to do so.

2. The time sharing mode, in which the camera will obtain a matrix from the scene when commanded to do so, as before, but it will continue to refresh this matrix with every scan of the camera maintaining the original parameter settings after it has returned from the camera routine.

The VDU must be used in a time sharing mode. One dedicated location in core is used to point to the area to be displayed. A routine in the executive ensures that the correct data word is sent to the VDU when it is requested. Sometimes it is advantageous to run the system without any time sharing peripherals. When this is required an option in the main program may be set which inhibits the VDU.

6.2.5 Special executive for ADMOS system.

A special version of the executive has been written which links with a general purpose fully time shared operating system 'ADMOS'.

In general, however, this version is not used very often for two main reasons. Firstly, the operating system uses more highly organised data structures than the usual executive which limits the effects of some of the options that are normally available to the system. E.g. when using the operating system one cannot stop the program to look at what is happening due to the time sharing. Secondly, the operating system requires 4K words of store (10,000 octal locations) which is far more than the normal executive and means that the data workspace available is greatly reduced.

The one advantage with this system is that the user can access the disc with the operating system which may be useful when much bulk data handling is required.

6.3 THE MAIN PROGRAM.

6.3.1 General outline.

The main program allows a user to access the data workspace which is divided into blocks of 16-bit words, it is 1,792 (3,400 octal) locations long. The on-line functions available to a user in the main program may be divided into three sections. 1. Camera control, 2. Data manipulation, 3. General purpose functions. These functions are described below. In the descriptions the following convention has been used

p	represents a pattern number
n	represents a number other than a pattern number.

Almost all of the functions are actually conducted by library subroutines.

6.3.2 Camera control functions.

A set of commands enable all the parameters sent to the camera to be varied. These commands are as follows.

RI	n	move right*	These commands allow the viewing window to be moved n steps in any direction.
LE	n	move left*	
UP	n	move up*	
DN	n	move down*	

* This indicates the command is sensitive to the global reset command RS which is described later.

ZO	n	Change the zoom value by n*
ZC	n	Change the zoom value by n* and maintain the position of the center of the viewing window. This changes the X and Y coordinates as they specify the top left-hand corner of the viewing window.
TH	n	Change the light threshold by n levels*
TS		Set the average threshold* to 1 after the zoom value has been output. This makes the hardware averaging logic very sensitive to any picture elements which are 1.
SS		(Show Status). This prints on the command device, the X address, Y address, zoom value and threshold setting.
OD		(Origin Defined). A predefined set of values for the above four parameters are output to the camera.
DO		(Define Origin). This command allows the above origin to be defined.
IH		(Eye Inhibit). Set the camera in the non-time shared mode.
IE		(Eye Enable). Set in time shared mode.
DG	n	Set an n milliseconds delay* after a matrix has been obtained from the camera.
IM	p	Move the matrix obtained from the camera to pattern p.

6.3.3 Data manipulation functions.

The commands which manipulate the patterns in the data workspace may be divided into smaller groups as described below

(a) Inputting patterns

IP p	(Input pattern)	Input a pattern from data channel (paper tape format)
RE p	(Read pattern)	Input a pattern

(b) Editing patterns

AL p n_1 n_2	(Alter pattern)	Put the number n_2 into row n_1 of pattern p.
CB p n_1 n_2	(Change bit)	Complement bit n_2 on row n_1 of pattern p.

(c) Manipulating a pattern

CP p	(Clear pattern)	Set pattern p to all 0.
FP p	(Fill pattern)	Set pattern p to all 1.
IV p	(Invert pattern)	Complement pattern p.
MK p n	(Mask pattern)	Set the first n bits of pattern p.
CR p	(Clockwise rotate)	Rotate pattern p by 90^o.
AR p	(Anti-clockwise rotate)	
RU p n	(Roll up)	Roll toroidally the pattern p by n rows or columns.
RD p n	(Roll down)	
RL p n	(Roll left)	
RR p n	(Roll right)	

(d) Operations of one pattern on another

IC p_1 p_2 (Interchange)

AN p_1 p_2 (AND) Pattern p_1 is ANDed with pattern p_2. The resultant pattern is in p_2

OR p_1 p_2 (OR)

XO p_1 p_2 (Exclusive OR)

HD p_1 p_2 (Hamming distance) Print the Hamming distance between p_1 and p_2 as a number and as a percentage.

MP p_1 p_2 (Move pattern) Move pattern p_1 to p_2.

(e) Output a pattern

DP p	(Display pattern)	Display pattern p on the VDU.
OP p	(Output pattern)	Output a pattern on the data channel in paper tape format.
PP p	(Print pattern)	Print pattern p with X representing a 1 and space representing a 0.
PB p	(Print binary)	Print pattern p as 16 binary numbers.
PO p	(Print Octal)	Print pattern p as 16 binary coded octal numbers.
LP	(List patterns)	This command causes an entry to a general purpose pattern dumping subsystem. A block of patterns either 2 or 4 side by side may be dumped by any one of the four possible output methods. The characters to represent 1 and 0 may also be defined.

DC $n_1 n_2$	(Display core)	Core locations n_1+n_2 to n_1+n_2+15 are displayed on the VDU.
PV	(Punch Visible)	Allows a visible tape heading to be output on the data channel.
TV		Allows a comment to be typed on the command device.
TP		Allows a heading to be output to the line printer.
SN	(Set net)	This command enters a subsystem to allocate some data workspace for simulated learning elements. The SLAM simulators are then (at run time) only allowed to access this area.
ID	(Inhibit display*)	This inhibits the VDU so that the system may be run without any time sharing.
RS		This command resets the next resettable command that is input. Resettable commands have been marked by *. E.g., RS LE will set the X coordinate of the viewing window to 0.

There is a global command SQ n (sequence) which repeats the next data manipulation function n times incrementing the pattern number each time e.g. 'SQ10 CP 1' will clear patterns 1 to 10. For functions of type (d) where two patterns are involved, both these pattern numbers are incremented.

There is a second global command for functions of type (d) SH n (Sequence with Hold) this command is also for repeating a function n times but in this case only the second pattern number is incremented.

6.3.4 General purpose functions.

This section is concerned with all the functions not covered in the previous two sections. The most important of these are described below

CM, PM, TC and RC have the same effect in the main program that they have in the executive.

PL	(Punch leader)	Output 100 blanks on the data channel.
FF	(Form feed)	Space to next page on line printer.
VS p_1 p_2	(View sequence)	This allows a sequence of patterns to be displayed on the VDU.
SD n	(Set Delay)	This allows a delay of n milli-seconds to be set which occurs after each pattern pointed at by the VS command is displayed.
DI	(Display eye)	Point the VDU at the camera input buffer.

6.4 The Experiment Subsystem.

6.4.1 Organisation of the main experiments.

During this project many different experiment subsystems have been written, the main ones are listed below. How these may be linked (by dumping the results from one and inputting them with the next set) is shown in Fig. 6.4.1.

Tracking Experiments.

All the tracking experiments have been conducted with two tracking subsystems: experiment 11 and experiment 12. Experiment 11 is for experiments with the maximum response tracking system described in section 3.3. The learning networks may be organised in Minerva or by simulation. Experiment 12 is for experiments which use only two learning networks to track with as described in sections 3.4 and 3.5. Both probability SLAM learning networks and CL SLAM learning networks may be simulated. Minerva may also be used for the probability learning network case.

Classifying Experiments.

The classifier experiments have been written to interact with the tracking experiments. Hence, this subsystem can receive information from the tracking system as it tracks and a classification can be made in real time. However, when developing the classifier, to be able to repeat exactly the same tracking motions many times was considered important. To achieve this Experiment 13, a tracking simulator was written. This subsystem stores the logged tracking motions from an actual tracking system

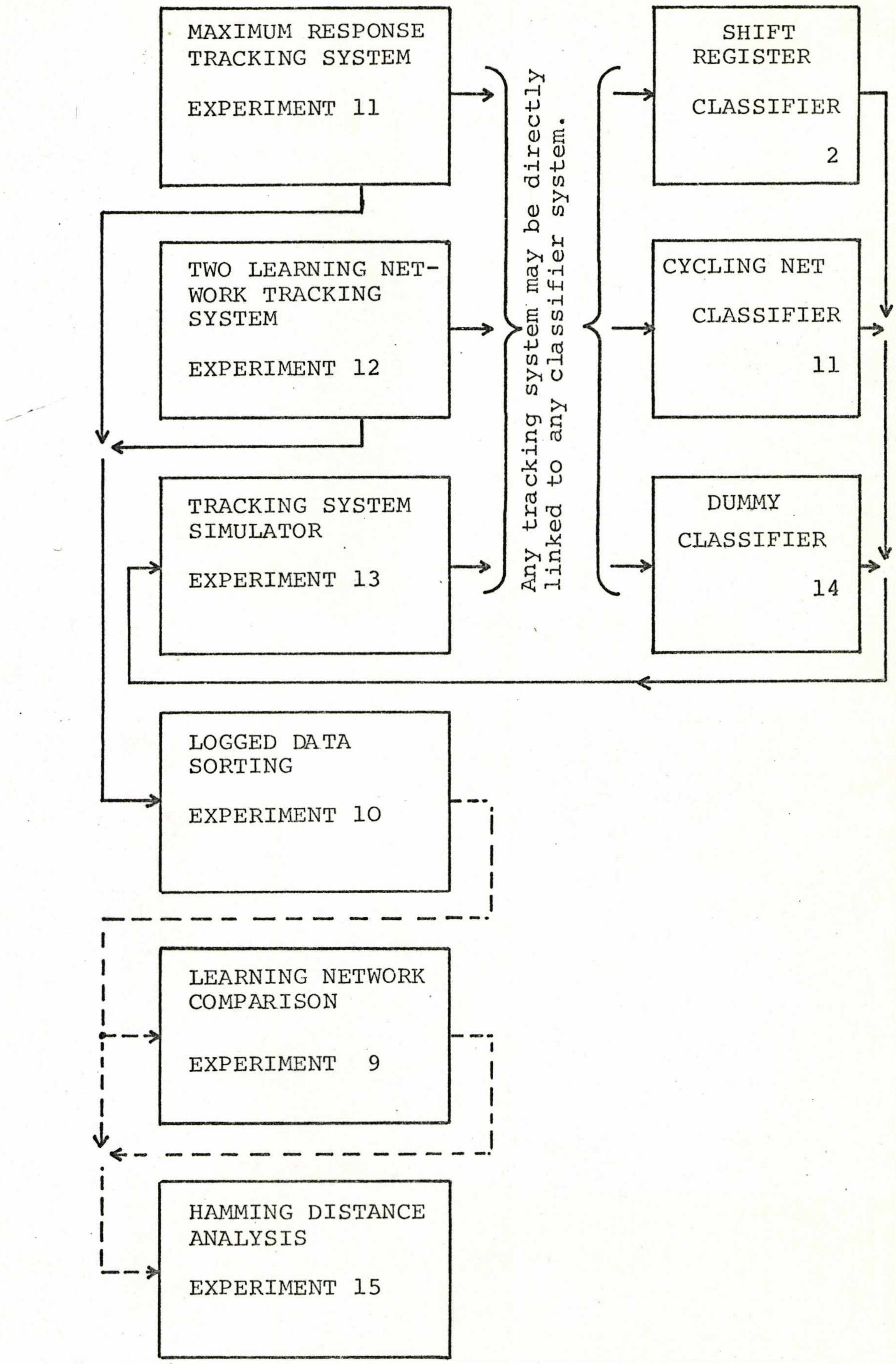

MAXIMUM RESPONSE TRACKING SYSTEM
EXPERIMENT 11
TWO LEARNING NETWORK TRACKING SYSTEM
EXPERIMENT 12
TRACKING SYSTEM SIMULATOR
EXPERIMENT 13
LOGGED DATA SORTING
EXPERIMENT 10
LEARNING NETWORK COMPARISON
EXPERIMENT 9
HAMMING DISTANCE ANALYSIS
EXPERIMENT 15
Any tracking system may be directly linked to any classifier system.
SHIFT REGISTER CLASSIFIER
2
CYCLING NET CLASSIFIER
11
DUMMY CLASSIFIER
14

FIGURE 6.4.1

experiment and can repeatedly present this information to a classifier sub-subsystem. The final developed version of the classifier, described in section 4.2, is conducted by Experiment sub-subsystem 11. The shift register classifier, mentioned in section 4.4, is conducted by Experiment sub-subsystem 2.

A problem with using the tracking system and classifier together is that with all their options they both require a lot of store. There is not enough room, for example, for the experiment 11 or 12 tracking system, the experiment sub-subsystem 11 and the backing system. However, by reducing some of the options in either of these will create enough room. Experiment sub-subsystem 14 was written to simulate the effect of a classifier for a large tracking system. It enables the tracking motions to be dumped (for use by experiment 13) and also allows the tracking motions to be plotted in real time on the graph plotter.

Other Experiments.

Experiment 9 is designed to test different learning networks simultaneously so that their performances can be compared. This experiment is used in section 5.6 and it is used as an example of a subsystem in section 6.4.2.

There are some other special purpose subsystems for example, Experiment 10 and Experiment 15. The tracking systems are able to dump all the patterns obtained from the viewing window and also other parameters (e.g. the responses from the learning networks) while it is tracking a pattern. For efficiency, these are dumped on paper tape in pattern data format. Experiment 10 is designed to sort out these patterns into a logical order and print the data in a readable form.

Experiment 15 is designed to obtain a Hamming distance distribution from sets of patterns. It can calculate a distribution for the Hamming distance between all the combinations of pairs of patterns within a set of patterns and it can calculate a distribution for the Hamming distance between all the combinations of pairs of patterns between two sets of patterns.

6.4.2 An example of an experiment subsystem.

Experiment 9 is designed to compare the performance of different learning networks. On the following pages an example of how this would appear to the on-line user is given. Experiment 9 has been used for the experiments in section 5.4 and its organisation is described there.

In this subsystem, all the operations are carried out from the subsystem command status. The functions which are available in this command status are listed below.

CH	(Change)	This command allows the chosen different types of learning networks to be selected and the store for them allocated.

SS	(Show Status)	Prints the selected learning networks and all the store allocations.
CN	(Clear Nets)	Sets all the learning network stores to 50%.
T+(or Tl)p	(Teach 1)	Teach pattern p with 1 on the teach sense terminals.
T-(or TO)p	(Teach O)	Teach pattern p O.
OR p	(Output Responses)	Print the responses obtained when presenting pattern p to the learning networks.
OP p	(Output Percentages)	Print responses but as percentages.
OH	(Output Heading)	For responses or percentages.
OD	(Output Device)	The output devices which are used by the above three functions may be selected with this command.
OS	(Output Sum)	Output the sums of all the previous responses. (This is used to find average responses).
RS	(Reset Sum)	Set all the summed values to zero.
SD	(Sum Device)	Allows the output devices for the sum values to be selected.
HE	(Heading for Sums)	Print a heading for sum values.
IM	(Input Maps)	This inputs the connections for the input mapping from the data channel.

The following learning networks may be simulated with the subsystem.

Simulated Learning Networks

N	Normal SLAM Learning Network
CC	4 Layer Cumulative Learning SLAM Network
TR	Ternary SLAM Learning Network
PB	Probability SLAM Learning Network
BC	16 Layer CL SLAM Learning Network
GP	N Layer Probability CL SLAM Learning Network

Minerva Learning Networks

N	Normal SLAM Learning Network
PB	Probability Learning Network

In the example shown in Fig. 6.4.2 two classes of handwritten characters of four patterns each are used as data. Usually, many input patterns are involved in an experiment and to control them all on-line by the user would be impractical. The set of commands to conduct such an experiment can be generated by a short simple program in a high level system such as Basic or a Macro processor. This string of commands is dumped on paper tape and the experiment can then be run under 'Tape Control'. The experiments described in section 5.6 were conducted in this way.

Comments

(allocate stores for the simulated SLAMs)

```
**  && SN
NO OF CARDS IS? 400
PATTERNS REQUIRED =100
NO OF SLAMS IS 1600
NO OF M I/P PATTERNS IS 25
MIN STARTING PATTERN IS 347
STARTING PATTERN IS ? 300  TO 399
& *  EXPERIMENT 9B
```

(Select and Set up the Learning Networks)

```
* CH  CHANGE
SIMULATION NET TYPES
 N  Y CL  Y  TR  Y  PB  Y  BC  N  GP  Y
STORE ALLOCATED  O.K.
1ST PATTERN USED IS 290
MINERVA NET TYPES
 N   N  PB  N
 PB  PROBABILITY/16 IS 4
 GP  PROBABILITY/16 IS 16
 GP INCREMENT SIZE IS 1024
 LEVELS = 63 LAYERS = 6
```

} (The GP SLAM Network is simulating a 6 output CL SLAM Learning Network)

(Obtain the settings and store allocations used in this experiment)

```
* SS  STATUS
MAP PATTERNS ARE 00290 TO 00297
I/P PATTERN IS 298
MAPPED PATTERN IS 299
NET STORE IS
 N  00300 TO 00303
 CL 00304 TO 00319
 TR 00320 TO 00327
 PB 00328 TO 00331 PROBABILITY/16 IS 4
 GP 00332 TO 00395 PROBABILITY/16 IS 16
INCREMENT SIZE IS 1024
 LEVELS = 63 LAYERS = 6
```

(Input the data and type a comment)

```
* IM   INPUT MAP PATTERNS   O.K.
* && SQ = 4 & IP 1 & SQ = 4 & IP 5 & TV
PAT'S 1 TO 4 CONTAIN 3'S, PAT'S 5 TO 8 CONTAIN 8'S.
                                                          #
```

(Display the input pattern on the VDU)

(Find the number of bits set in pattern 1)

```
& DP 298
& CP 0 & HD 1 TO 0
00062  00024%
```

```
& *  EXPERIMENT 9B
* CN
* OH          N       CL       TRD      TRV      PB        GP
* OP  1     00050    00050    00050    00000    00050    00050.00000
* T+  1
* OP  1     00100    00057    00100    00100    00062    00051.56254
* OR  1     00064    00512    00064    00064    00040    0002162624
* RS   SUM DELETED
* OP  1     00100    00057    00100    00100    00062    00051.56254
* OP  2     00081    00054    00100    00062    00056    00050.97658
* OP  3     00060    00051    00100    00021    00051    00050.34180
* OP  4     00070    00053    00100    00043    00053    00050.68361
* OS          00311    00215    00400    00226    00222    00203.56455
```

(Threshold for average sum is 4x50 = 200)

```
* RS   SUM DELETED
* T-  5
* OP  1     00062    00054    00100    00062    00051    00050.97658
* RS   SUM DELETED
* OP  2     00050    00052    00088    00039    00048    00050.46387
* OP  3     00046    00050    00057    00021    00048    00050.04882
* OP  4     00050    00051    00080    00032    00046    00050.31739
* OS          00146    00153    00225    00092    00142    00150.83010
```

(Test set of patterns 2 to 4, average threshold for sum is 3x50 = 150)

```
* OH          N       CL       TRD      TRV      PB        GP
* OP  5     00000    00045    00000    00062    00039    00049.02340
* RS   SUM DELETED
* OP  6     00032    00049    00037    00042    00045    00049.82909
* OP  7     00037    00049    00031    00025    00045    00049.85350
* OP  8     00037    00050    00055    00028    00042    00050.04882
* OS          00106    00148    00123    00095    00132    00149.73143
* OH          N       CL       TRD      TRV      PB        GP
```

(Print the data set of 3s using the pattern dump subsystem)

```
* && LP  READY?  N
 TT
 CC  0= . 1= X
 RE READY  FROM PATTERN 1  TO 4

.....XXX........  ................  .....XXXXXXXX...  ................
...XXXXXX.......  .....XXXX.......  ....XXXXXXXXX...  ....XXXXX.......
...XXXXXXX......  ...XXXXXX.......  .....XX.XXXXX...  ...XXXXXXX......
...XX...XX......  ...XXXXXX.......  .........XXX....  ...XXX..XXX.....
........XX......  ......XXX.......  .........XX.....  ...XX....XX.....
.......XXX......  ......XX........  ........XXX.....  .........XXX....
......XXX.......  .....XXXXX......  ........XXXXX...  ........XXXX....
.....XXXXX......  ......XXXXX.....  .........XXXX...  .........XXXX...
......XXXXX.....  .........XX.....  ............XX..  ..........XXX...
.........XX.....  ..........XX....  ............XX..  ...........XX...
..........XX....  ...........XX...  ............XX..  ...........XX...
..........XX....  ...........XX...  ....XX......XX..  ....X......XX...
..........XX....  ..........XXX...  ....XX......XX..  ....XX...XXX....
....X....XXX....  .........XXX....  ....XX.....XXX..  ....XXXXXXXX....
....XXXXXXX.....  .....XXXXXX.....  .....XX..XXXX...  .....XXXXXX.....
.....XXXXX......  ......XXXXX.....  .....XXXXXXX....  ................

  O.K.
& PL
```

Characters underlined are input by the user.

FIGURE 6.4.2

CHAPTER 7

CONCLUSION

In this thesis a pattern recognition scheme has been developed which tracks patterns and classifies them. The main feature of the scheme is that both of the above functions are achieved by adaptive learning networks. All the systems described may be easily realised in hardware. In general, the tracking system requires 256 16-bit words (4,096 bits) of active store and the classifier requires 64 16-bit words (1,024 bits) of active store.

This work represents only the initial researches on this type of system and a practical form of this pattern recogniser would require further development. The development of the tracking system, classifying system and learning elements will be reviewed in the next three sections.

7.1 THE TRACKING SYSTEM.

The details of the tracking system are given in Chapter 3. Initially the tracking system based its decisions on the localised information from the area of attention only. It has been shown that such a system may easily be taught to track edges of patterns but it is not sufficient to track the lines of patterns.

In order to track lines, feedback from the output to the input of the tracking system has been added to good effect and this has been further improved by a damping process. With this tracking system it is possible to track simple line drawings. For very simple patterns such as circles and triangles, the system behaves very well.

Investigations with this system have shown the following:-

1. Not all line drawings can be tracked, there are theoretically determined limits of the system (defined in section 3.2.4) and any complex scanning path must be considered carefully.

2. The performance of the system is very dependent on the ability of the teacher.

3. It is more difficult to teach complex patterns* to the system than simple ones.

4. As the patterns become more complex the generalisation rapidly deteriorates. (Increasing the size of the learning networks does not improve the generalisation.)

The first characteristic of the system mentioned above is due to the overall structure of the tracking

* The complexity of a pattern is difficult to define rigorously. In this case it is related to the number of line junctions and also to the nature of the line junctions (e.g. the number of lines entering the junctions and the relative shapes of the junctions).

system (i.e. the way that feedback is applied). The other three characteristics are determined by the nature of the learning modules. Three different types of n-tuple learning elements have been investigated and the degree to which they exhibit these characteristics has been detailed in Chapter 3.

The tracking system is very simple in structure and there are many ways in which it could be further developed. It cannot track all possible line drawings with only a short-term memory (created by the feedback). However, if a long-term memory (storing the sequence of features or tracking movements encountered) could be added, then there is no theoretical limit to the complexity of the patterns which can be tracked.

One possible development would be to select only a fraction of the learning network for tracking or testing. The fraction selected would depend upon the location of the viewing window on the pattern e.g. it could depend on either the position of the viewing window or the last direction taken (averaged over several steps). This development could also be applied to the classifier.

It would be interesting to see if a model of the system proposed by Noton(34) could be developed in which the classifier stores attention shifts necessary to examine the pattern and the tracking system could be guided at critical points by feedback from the classifier.

Analysis of the patterns taught to the learning modules has shown that the separation between classes is very small in Hamming distance, whereas the variation within the classes is very large. This could be improved by the introduction of some preprocessing. There are many different types of preprocessing which could be applied to the information from the camera. For example, the effect of processing the information with a layer of simplified models of retinal ganglion cells could be investigated e.g. with similar receptive fields to those described by Rosenberg and Wilkins(40). This could be further developed by adding layers of cell models which have similar properties to the cells found by Hubel and Wiesel(24) in the cats visual cortex.

So far no heuristics have been built into the processing of the information with regards to lines. If the task desired of the system is limited to tracking lines, then feature extractors may be applied to the input matrix to extract information which is relevant to line drawings only. For example, one could extract the number of lines in the window, their orientations, their positions with respect to the center of the window, etc. Hence, the relevant information would be presented in a less redundant form to the system and one might see an improvement in the performance of the system.

7.2 THE CLASSIFIER SYSTEM.

The details of the classifier system are given in Chapter 4. For the classifier an attempt has been made to use a learning network to classify a sequential string of input data. Although this task has often been tackled with algorithms (e.g. all the tracking methods described in section 1.2.4), an approach with learning networks has not been tried before.

A simple system has been tentatively proposed and developed to achieve this task but the results from it have been poor. This classifier only receives input information from the tracking motions of the tracking system and is designed to generate a codeword to indicate the class of the input sequence.

Hence, with this approach, the tracking motions must contain the information to classify the input pattern. In this case, the tracking system tracked the lines or edges of the input pattern but this is not the only scheme with which the classifier could be used. For example, it could try to classify "saccadic" motions provided that they be unique for each class of patterns.

The system which was finally developed consisted of two parts: a cycling network, which is stimulated by the input sequence and a codeword extractor which detects classifying states in the cycle network. The main problem here is to know what to teach to the cycle network and very little is known about methods of teaching such a system. Also, it is very difficult to analyse the results of teaching due to the sequential nature of the input and output.

To assist with the study of this problem, the concepts of SLAM store 'penetration' and SLAM store 'overlap' have been introduced. Although in such an undeveloped form their usefulness is still questionable, they do allow measures to be made on the input data and internal states of the classifier which are independent of the sequential nature of the input data and its associated effects,* (e.g. the initial state transient). An important feature of patterns associated with the classifier is the frequency of occurrence of elements of these patterns during the input cycle and neither of the above concepts embodies this. To provide a measure for this feature, the concept of pattern activity has been introduced but this has not been rigorously defined and is used in a qualitative way only. The use of these measures is illustrated in detail in Chapter 4.

The work in this thesis has done little more than establish the basic foundations of a classifier of this type. The learning network investigated was very small (only 16 TR SLAM-16s were used for the feature extractor) and future work could be aimed at investigating the properties of a larger system.

---

* The main problem with these measures is that often important information is conveyed by the sequential order of the data.

It has been found advantageous to use T.R. SLAMs in the code net. Experiments with the cycle net have shown that the method used for its teaching has not been very effective and also cycling networks are very sensitive to any changes in their structure. Hence, a teaching algorithm which can only cause small amounts of change may work better. One possible scheme to realise this would be to allow each memory element in the learning network to be taught once only. (This is similar in basic concept to the origins of the T.R. SLAM and in practice could be realised by learning elements consisting of two SLAMs with their inputs commoned.) Another possibility would be to use T.R. SLAMs with the 'last valid output' used for the outputs of the net.

The method used to code the sequential tracking information before inputting it to the classifier is important (this is due in part to the small size of the learning networks used). Several methods have been investigated and the most useful form used was to input the position of the area of attention. There are many other possibilities which could be investigated as in the following examples.

1. The input could be applied so that it had an inhibitory effect on the feedback only. Hence, the input 'inhibits' rather than stimulates the net activity. This may overcome the problems of the dependance on initial starting and the length of the initial transient states.

2. In many pattern recognising schemes, localised information about features is often considered important. This information could be obtained by extracting features from the tracking data for the last few steps taken. Then this information could be input to the classifier in addition to the position information.

3. Another more powerful method of realising the above concept is to extract features directly from the area of attention and input these with the position information.

7.3 THE DEVELOPMENT OF THE LEARNING ELEMENTS.

Details of the development and properties of the learning elements are given in Chapter 5. The initial work was conducted with SLAM-16 learning elements. Three basic developments of this element have been investigated; the probabilistic SLAM, the CL (Cumulative Learning) SLAM and the TR (Ternary) SLAM.

The probabilistic SLAM element is similar in structure to a normal SLAM-16. However, on giving a teach command there is a definite preset probability that this command will be ignored. This teaching mechanism makes a learning network less sensitive to the last pattern taught and is ideal for cases where an ageing teach process is required,for example, the method used by Fairhurst[(14)].

The CL SLAM has been patented(39) and is characterised by having a number output, rather than a binary one, the value of which depends on the frequency of occurrence that the input pattern has been taught. This SLAM requires n times the amount of store for a normal SLAM where n is the number of bits at the output. (For most cases a value of n=4 has been used.) Like the probability SLAM network, the CL SLAM network overcomes the overteaching problem of the last pattern taught with the difference that it does not suffer from the defect of the probability SLAM network which ignores part of the input pattern on teaching to achieve this.

Since the development of the CL SLAM, a frequency sensitive SLAM (the FO SLAM) has been mentioned by Chung(7) This FO SLAM uses an internal algorithm to achieve a dynamic equilibrium with respect to the number of states filled within the SLAM which is considered to be an important feature. On the other hand, it is the contention with the CL SLAM that some outputs should be weighted more than others over the network. The FO SLAM could be realised with a CL SLAM structure and with a saving of store. A FO SLAM could be realised by an n output CL SLAM k which requires $n.2^k$ bits of active store; to achieve this with the shift register method by Chung would require $2^{n.k}$ bits of active store.

The normalisation technique described by Chung may be useful for some CL SLAM learning networks and the CL SLAM teach mechanism could easily be adapted so that a distribution of values is always maintained within the CL SLAM outputs.

The TR SLAM has a ternary output (1, 0 and not valid) and uses twice as much active store as a normal SLAM. The TR SLAM was developed initially for use at the output of cycling learning networks. (Also, the best results from the tracking system were obtained with learning networks which had a similar structure to a learning network of TR SLAMs.)

One feature of the TR SLAM is that it records all patterns it has been taught. This means that the last pattern taught has no more effect than any other. However, by the same token, these learning networks are easily overtaught.

Another feature of the TR SLAM is that its stores may be easily analysed for 'penetration' and 'overlap' and also, when being used, noting the number of valid outputs can be used to indicate a confidence level for the classification.

7.4 CONCLUDING REMARKS.

In addition to the investigation and development of a pattern recognition system, some special purpose hardware and software has been developed.

To obtain data from a visual scene a normal television camera has been connected to the computer via a hardware control unit which was built especially for this project. This hardware is orientated towards realising some of the features of the human eye in that it can only obtain detailed information from a small part of the scene at a time. However, it may also be used for general purpose data acquisition from a visual scene. It has the following specifications:

1. The scene is defined by a 256x256 square matrix which is viewed by the camera.

2. A 16x16 bit binary matrix representing an area of the scene within the viewing window may be obtained every 20 milliseconds.

3. The binary matrix is obtained by sampling the scene with one of 16 brightness levels.

4. The viewing window may be positioned anywhere on the 256x256 scene matrix and may cover an area of 16x16 picture elements or any multiple of this. (When the viewing window covers an area greater than 16x16 picture elements, a hardware averaging unit is used to generate the 16x16 bit binary output.)

A special operating system has been written for this project. It has the following features:

1. It contains routines to control the television camera and all the other peripherals including Minerva.

2. It allows on-line access and manipulation of a data workspace, organised in 16x16 bit binary patterns.

3. It contains routines to simulate learning networks of normal, probability, CL and TR SLAMs.

4. Each experiment is conducted by a subsystem which may be any DAP program.

5. An interacting debugging program may be loaded when required.

The operating system is not restricted to this project only but is designed to be useful for experimental work involving pattern processing in general.

An attempt has been made to establish a framework for and develop a pattern recognition system from basic logical considerations rather than basing it on any existing system. Further development would be aimed in one of two directions. The system could be developed towards a pattern recogniser by adding heuristic feature extractors etc. which are known to be useful for other systems. Alternatively, one could work towards a model of the eye and the visual perception mechanism found in man. Hypotheses of the visual processing mechanisms in the eye, for example the function of the ganglion cells, could be investigated with this system. Some of the possible developments of the tracking system are given in section 7.1. The classifier which has been developed is only one of many possible forms, suggestions for further developments of this and for alternative methods are given in section 7.2.

Whatever direction the further development may take, the hardware and software systems described here should be useful as a foundation. The existing system sets a reference with which to compare further results. Also, the development made in learning elements should be considered when designing future systems.

Appendix 1

Circuit Details Of The Camera Hardware

A description of the operation of the hardware control unit is given in section 2.2. In this appendix the circuit details of this hardware are given. A block diagram of the hardware modules is shown in Fig. A1.1, definitions of the interconnections are given in Table A1.1 and the circuits of the modules are given in Figs. A1.2 - A1.9. The symbols used in these circuits are defined in Fig. A1.10 and Fig. A1.11.

The following is a brief description of the functions of the modules and their interconnections.

The 6MHz Clock Fig. A1.2: The clock produces the timing pulses for setting up the 256x256 matrix over the scene and generating the camera synchronising pulses. Due to the poor quality of the camera a lot of mains hum was present on the video signal. To overcome the beating effects of the hum, caused by the difference in frequency between the frame rate and the mains, the clock was locked to the mains frequency by a phase locked loop.

A simple phase locked loop was originally used but, due to the large difference in frequencies (6MHz - 50Hz), the clock frequency varied an unacceptable amount during one cycle of the mains. This was overcome by a combination of three methods:-

1. The power supplies were heavily decoupled.
2. Two integrators instead of the usual one were used.
3. The oscillator was isolated from the rest of the circuitry by an enclosed metal box mounted at the other end of the rack.

The output of the clock is via CK and the inverse via $\overline{CK}$. A reference strobe for the phase locked loop when the Y counter is reset is provided by SC.

The X Counter and Decode Module Fig. A1.3: This module counts the clock pulses and determines the x coordinate of the television scan. The decoder generates the following functions:

LSY	Line sync pulse, to synchronise the camera. This is also used as a timing pulse for the zoom counters.
XPOS	Indicates when the x coordinate of the viewing window is reached. (Note XIC occurs slightly before XPOS and is used as a reset pulse by the line sampler.
SY	This resets the X counter and increments the Y counter.

The Y Counter and Decode Module Fig. Al.4: This module counts the cycles of the X counter and determines the Y coordinate of the television scan. The decoder generates the following functions:-

FSY	Frame sync pulse to synchronise the camera frame.
YPOS	Indicates when the Y coordinate of the viewing window is reached.
FF	This indicates to the computer when the frame flyback occurs.
SC	This resets the Y counter and strobes the phase locked loop of the clock.

The Zoom Input Buffer Fig. Al.5: This module enables the zoom value ZV5 (5 bits) and the average threshold ATH4 (4 bits) to be input. Usually the average threshold is half of the zoom value and provision has been made for this to be achieved automatically.

Video Processor Module Fig. Al.6: This module receives the composit video data from the T.V. camera and converts it to a binary signal BVI by means of a comparator. The threshold of the comparator is set at one of 16 levels by the computer or by the manual controls. This module also impresses the viewing window onto the composit video signal which is output to the monitor. The time when the window is to be impressed is defined by both zoom counters being active i.e., when XZA and YZA are true.

The X Zoom Counter and Line Sampler Fig. A1.7: This module samples the binary video input BVI when a line which intersects the viewing window is scanned. The moment to start sampling is indicated by XPOS. XIC changes state before XPOS and clears the data counter. Once XPOS occurs the X zoom counter counts the number of picture elements for each bit (defined by ZV5) and the data counter counts the number of picture elements set at 1. This latter count is compared with the average threshold ATH4, if it is greater or equal to the threshold the sampled data value SDAV is set true. After each bit of output data is obtained, it is strobed into the Line Average Module by SDAS. 16 bits of data are obtained in this way and then a pulse on LEND indicates the end of sampling. LSY resets this module at the end of every line. While the line is being sampled, XZA is set true.

The Y Zoom Counter Module Fig. A1.8: This module counts the lines which are relevant to the viewing window. The first line of the viewing window is indicated by YPOS. The Y zoom counter is strobed by LSY i.e., during the line flyback. After each zoom value number of lines (defined by ZV5) and when the line averaging is complete (indicated by DWC), 'word ready' WORR is set which initiates a data transfer to the computer (also, the average counters are cleared by a pulse on CAC). The computer indicates when the data has been accepted by a pulse on 'word accepted' WORA. If the response from the computer is too slow then 'Frame not valid' FNV is set. This is reset by the computer with a pulse on RFNV.* While the lines are being counted YZA is set true.

The Line Average Module Fig. A1.9: This module averages the data obtained from several lines. The data from the line sampler SDAV is strobed by SDAS into a 16 bit shift register. When line sampling is complete (indicated by a pulse on LEND), the contents of the shift register are used to increment 16 4-bit counters. Each counter registers the number of lines in which the line sample has indicated a 1 for that position. If the counter value is equal to, or greater than, the 'average threshold' ATH4, then the relevant bit in the output shift register is set. After each zoom value number of lines a pulse from the Y zoom counter on CAC resets these counters. The counters are compared serially with the average threshold, after the line has been sampled, by 4 16 to 1 bit multiplexers and a 4 bit address counter. When this averaging process is complete, a pulse is output on DWC to the Y zoom counter. The final 16 bit data word which is output to the computer is reset by the computer with a pulse on WORA.

* This feature was originally included as the camera hardware was connected to a standard 16 bit computer interface. A hardware 16x16 bit buffer has now been built into the interface, hence the reply from the computer is independent of the computer program and is always quick enough.

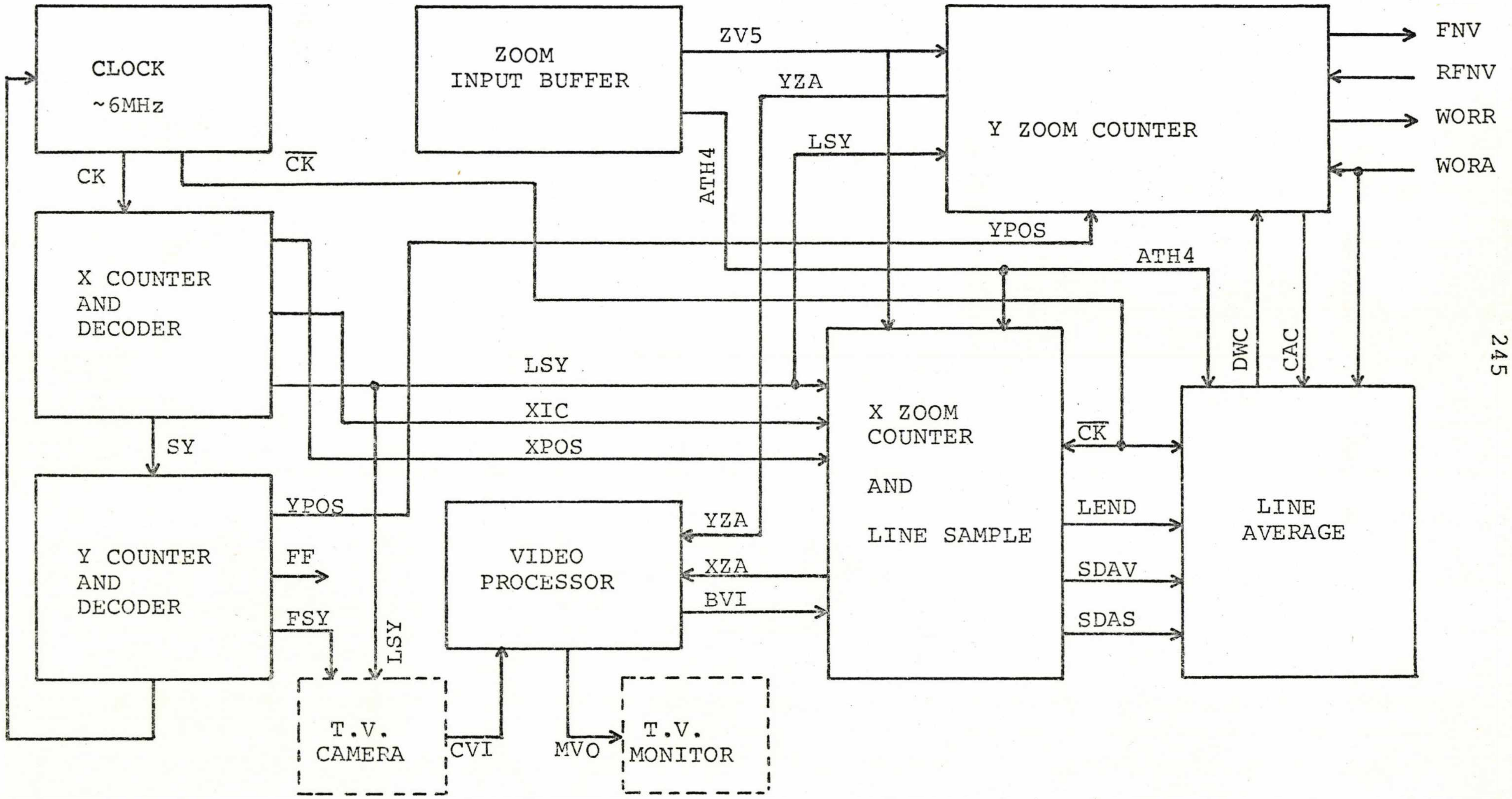

CLOCK
~6MHz
ZOOM INPUT BUFFER
Y ZOOM COUNTER
FNV
RFNV
WORR
WORA
ZV5
YZA
LSY
ATH4
CK
$\overline{CK}$
X COUNTER AND DECODER
YPOS
ATH4
DWC
CAC
LSY
XIC
XPOS
SY
X ZOOM COUNTER AND LINE SAMPLE
$\overline{CK}$
LINE AVERAGE
LEND
SDAV
SDAS
Y COUNTER AND DECODER
YPOS
FF
FSY
LSY
VIDEO PROCESSOR
YZA
XZA
BVI
T.V. CAMERA
CVI
MVO
T.V. MONITOR

Figure A1.1

Interconnection List

ATH4	Average Threshold (4 bits)
BVI	Binary Video Input
CAC	Clear Average Counters
CK	Clock
$\overline{\text{CK}}$	Clock Inverted
CVI	Composit Video Input
DWC	Data Word Complete
FF	Frame Flyback
FNV	Frame Not Valid
FSY	Frame Synchronising Pulse
LEND	Line End
LSY	Line Synchronising Pulse
MVO	Monitor Video Output
RFNV	Reset Frame Not Valid
SC	Strobe Clock
SDAS	Sampled Data Strobe
SDAV	Sampled Data Value
SY	Strobe Y Counter
WORA	Word Accepted
WORR	Word Ready
XIC	X Initial Clear
XPOS	X Position Reached
XZA	X Zoom Counter Active
YPOS	Y Position Reached
YZA	Y Zoom Counter Active
ZV5	Zoom Value (5 bits)

TABLE A1.1

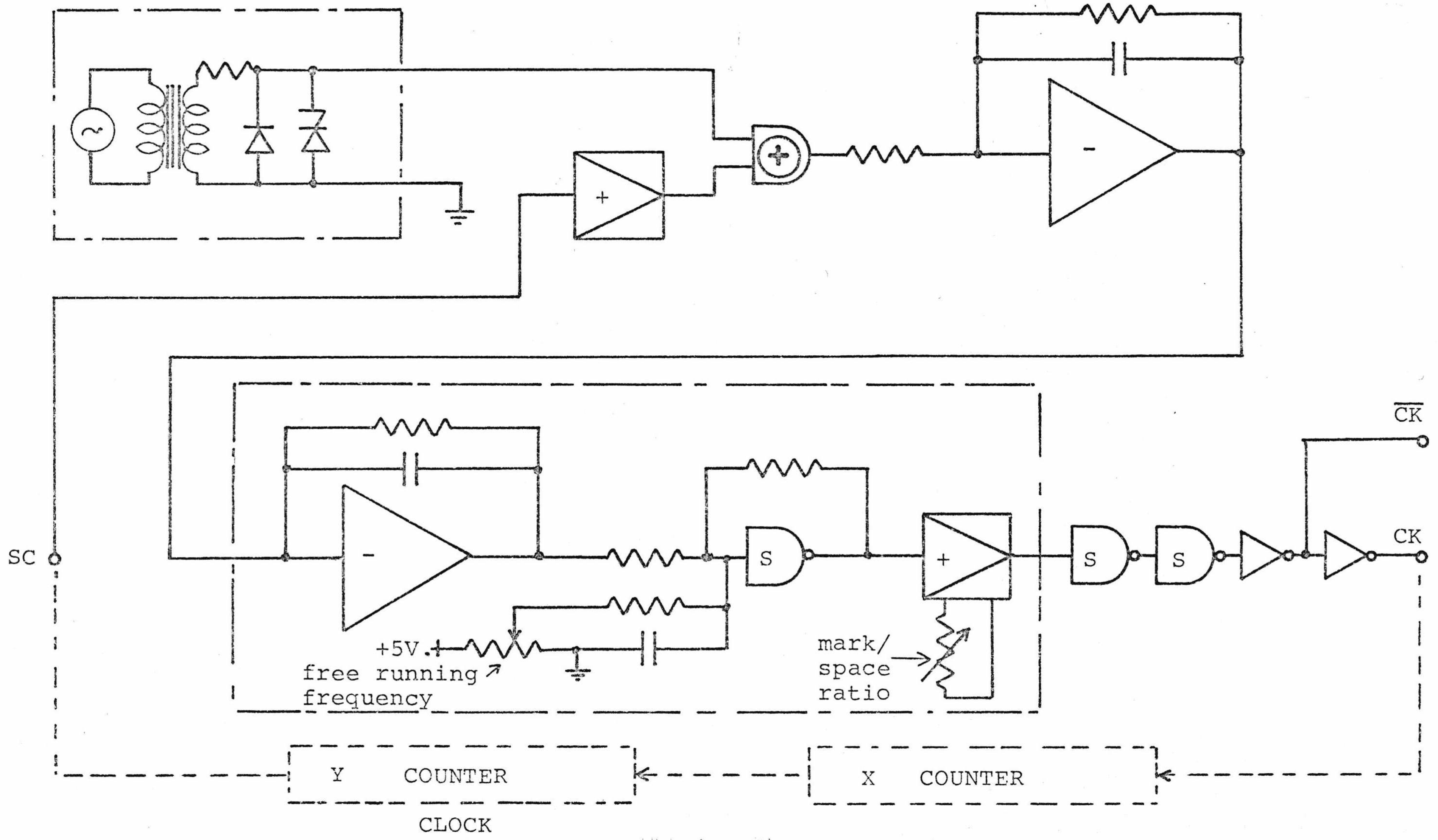
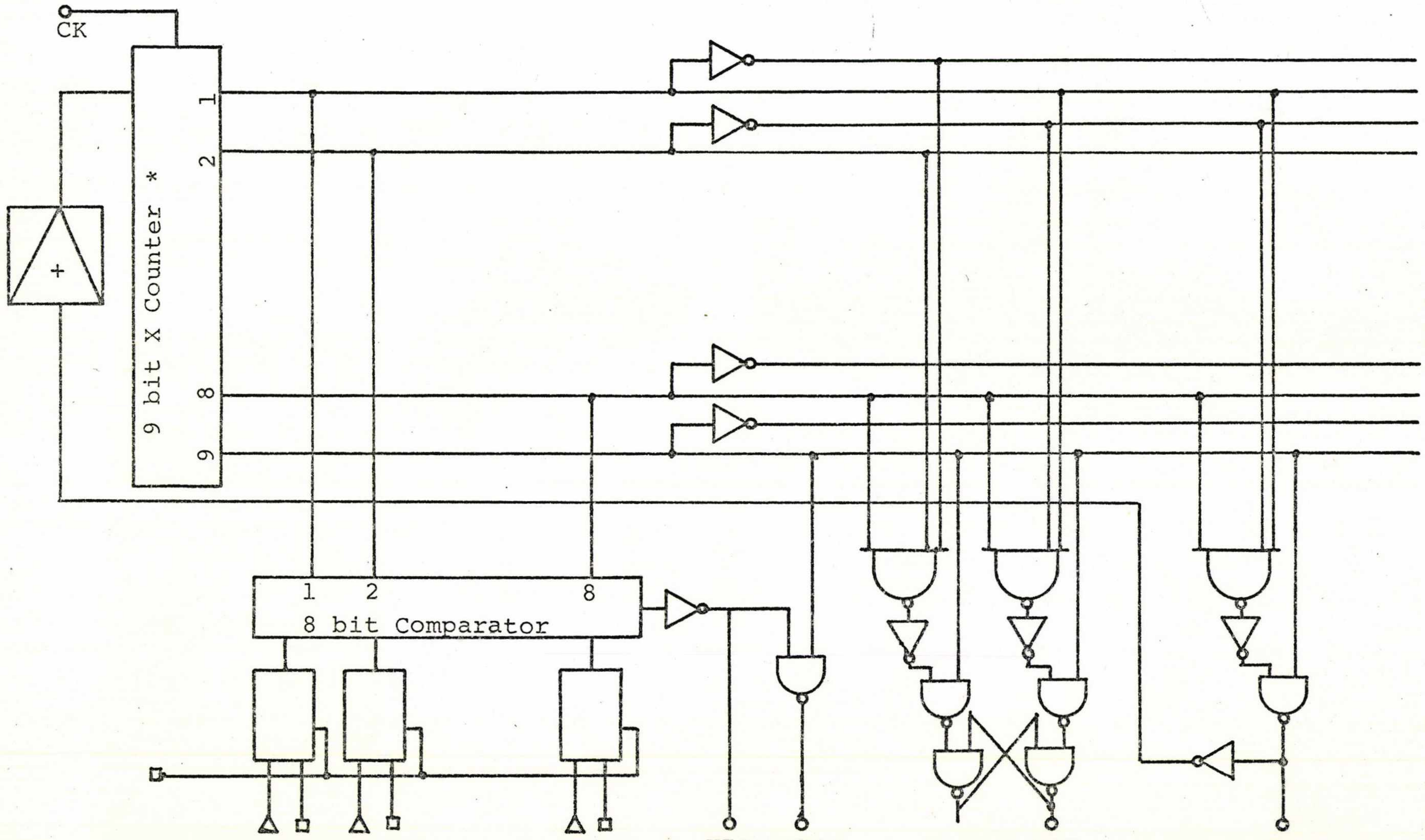
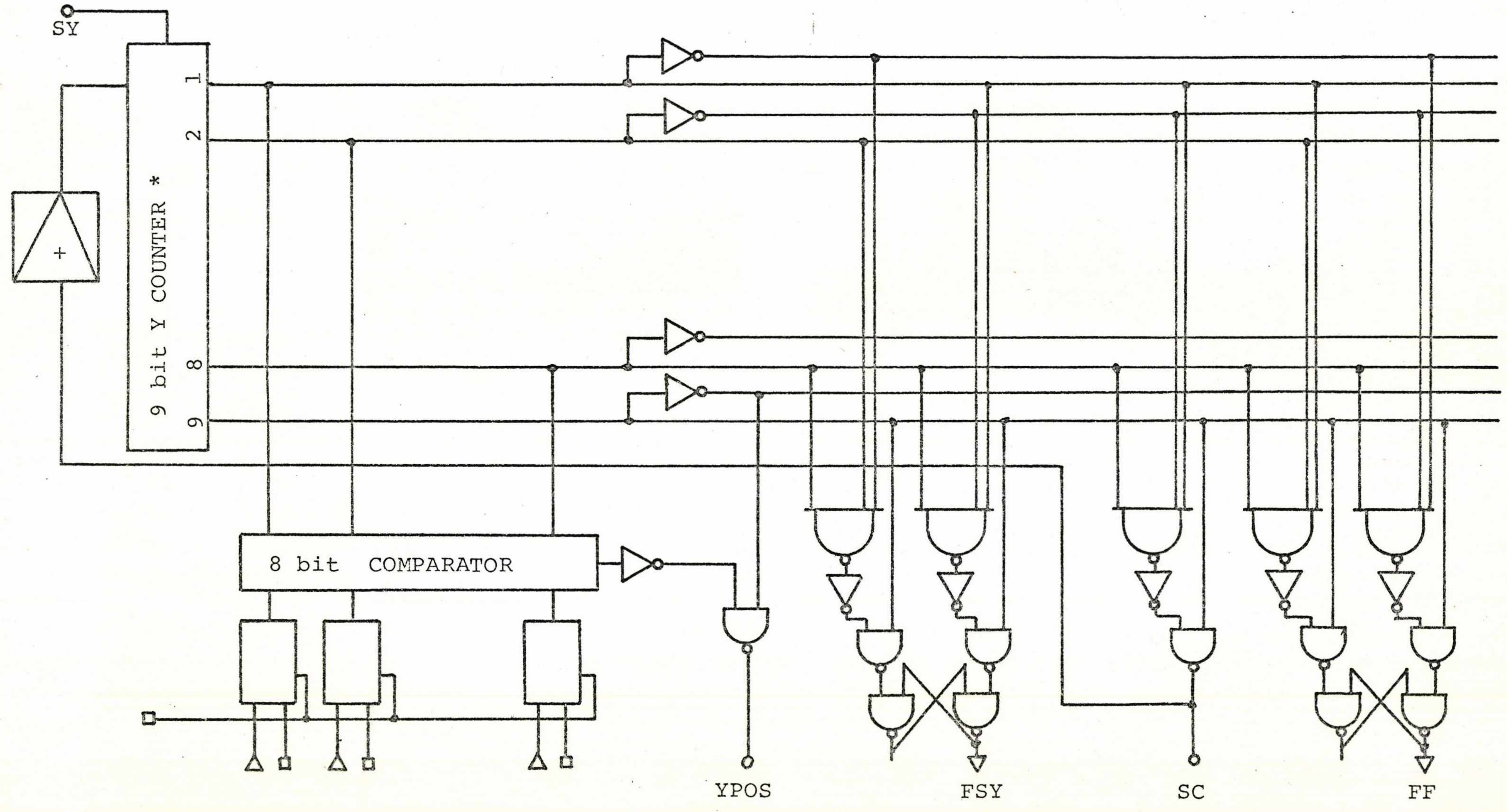

+
+
-
-
S
+
S
S
+5V.
free running frequency
mark/ space ratio
$\overline{\text{CK}}$
CK
SC
Y COUNTER
X COUNTER
CLOCK

FIGURE A1.2



X Counter and Decoder

FIGURE A1.3



Y COUNTER AND DECODER

FIGURE A1.4

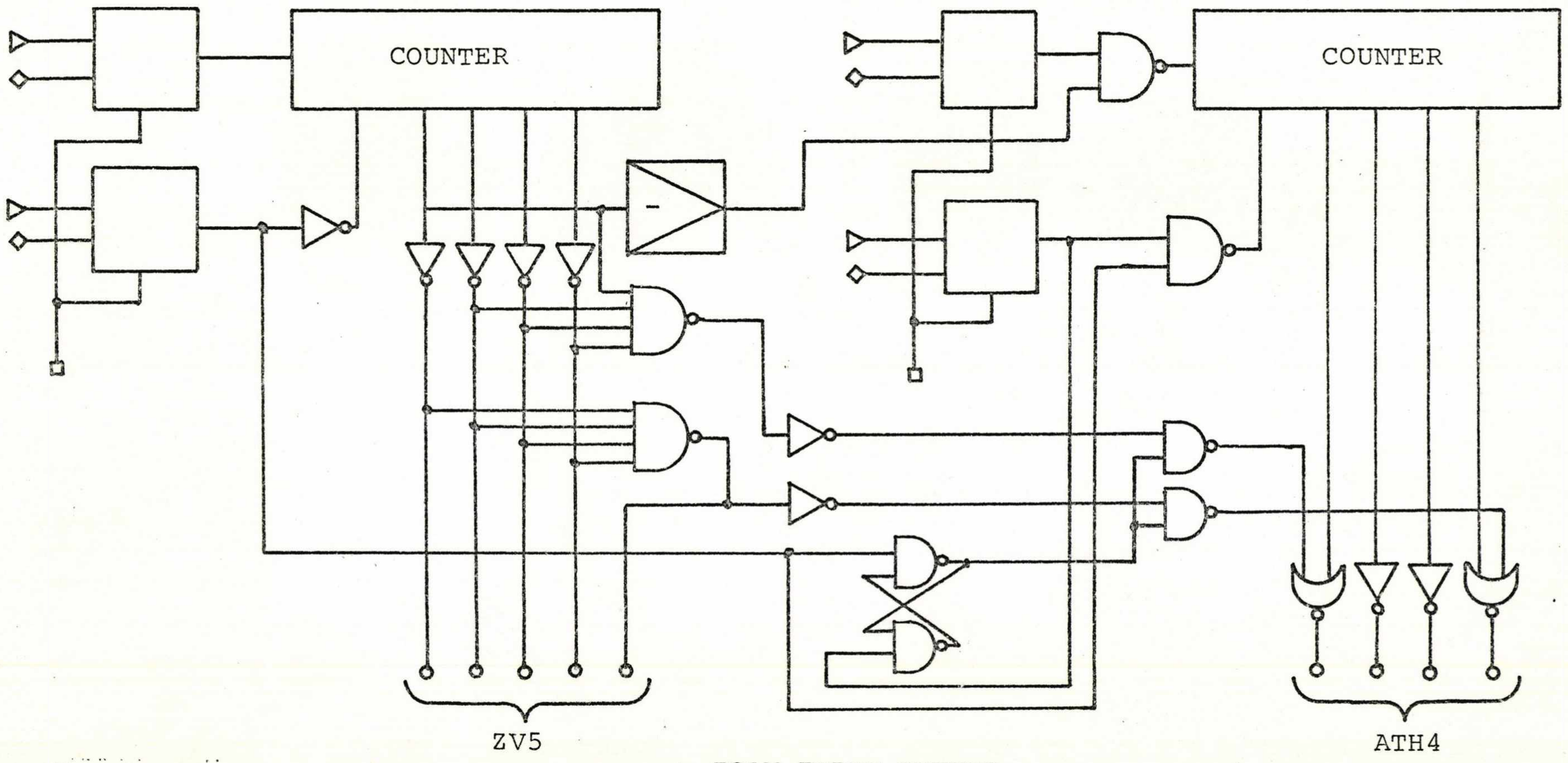
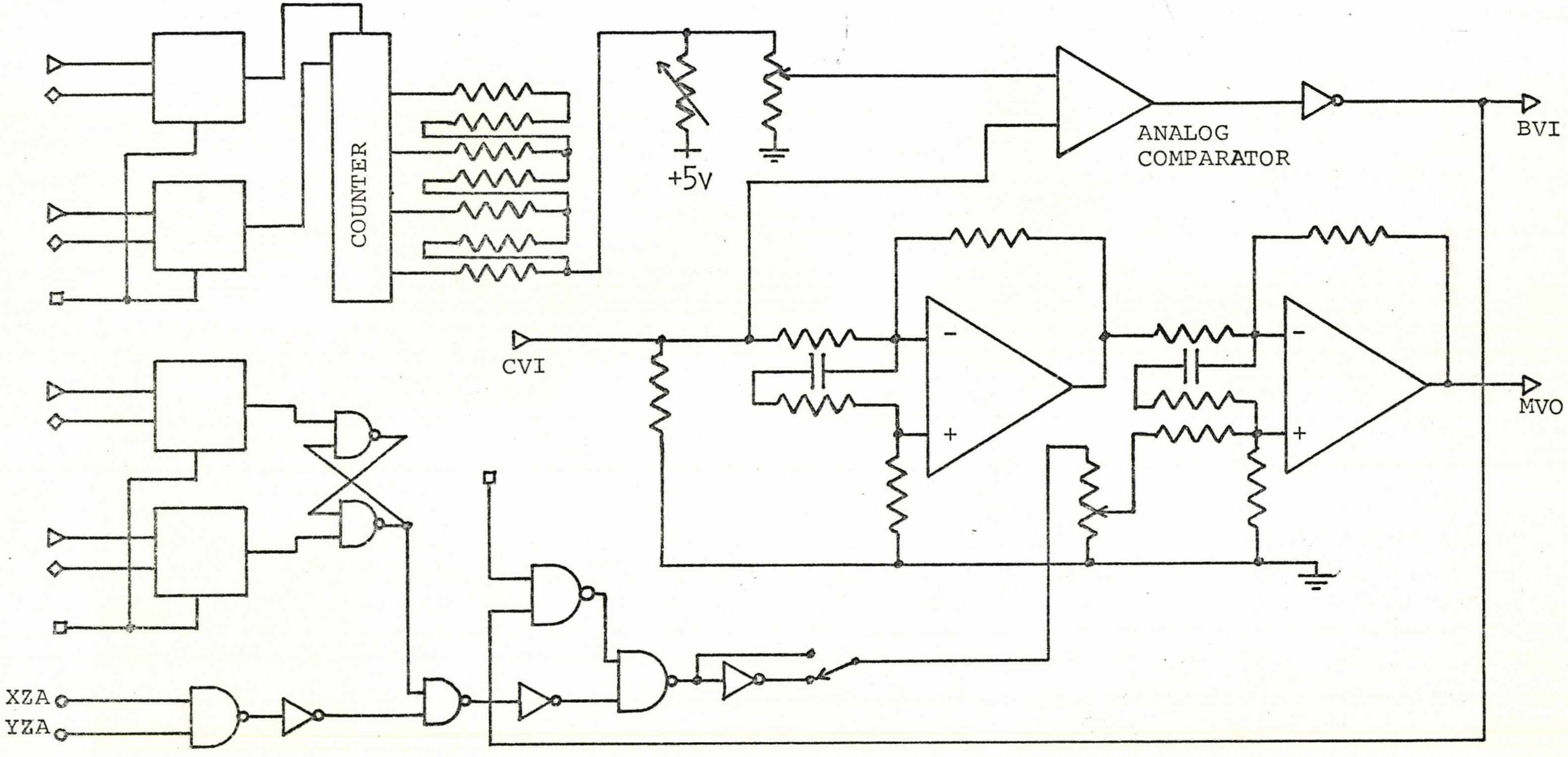
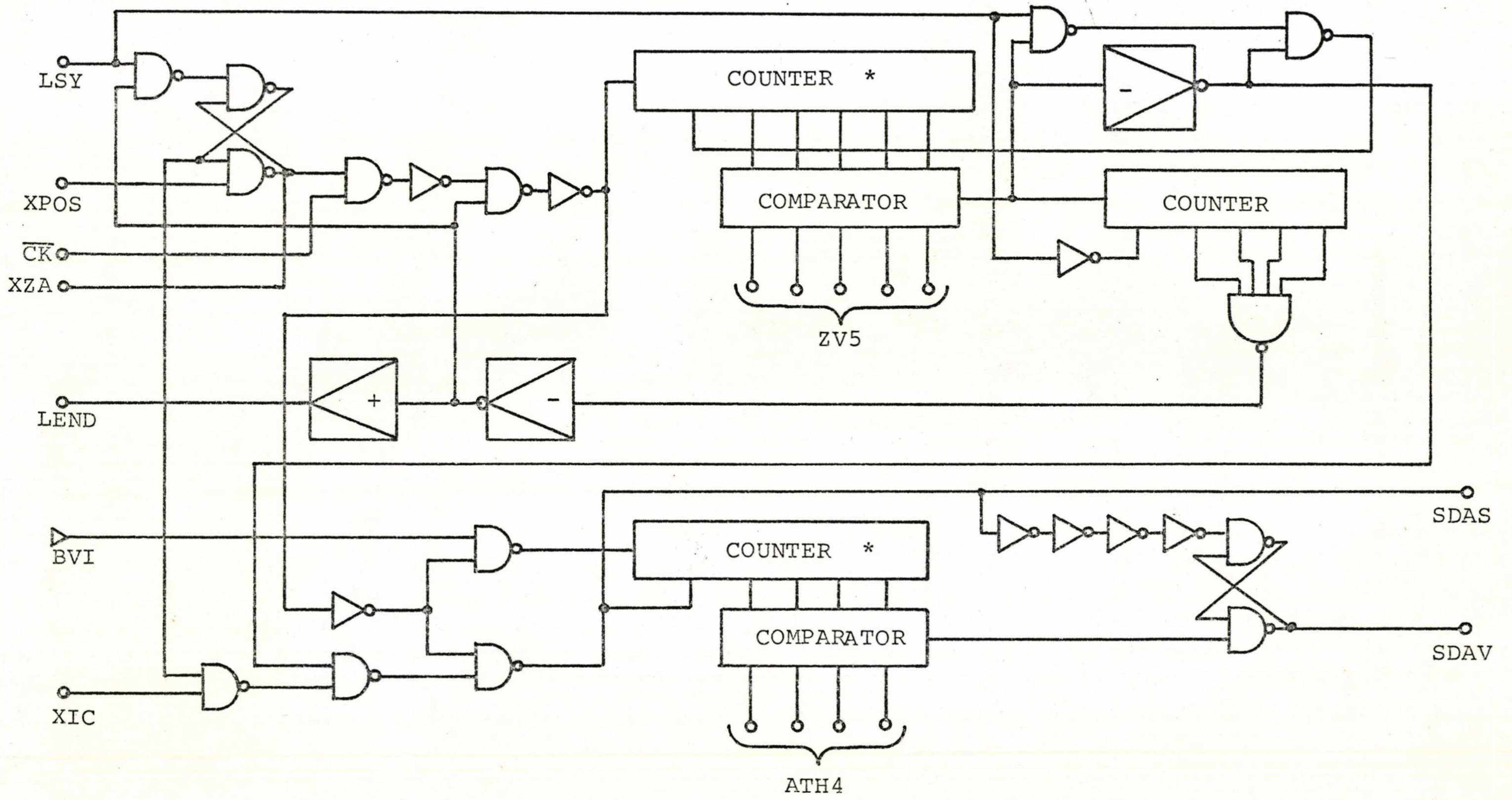


FIGURE A1.5 ZOOM INPUT BUFFER



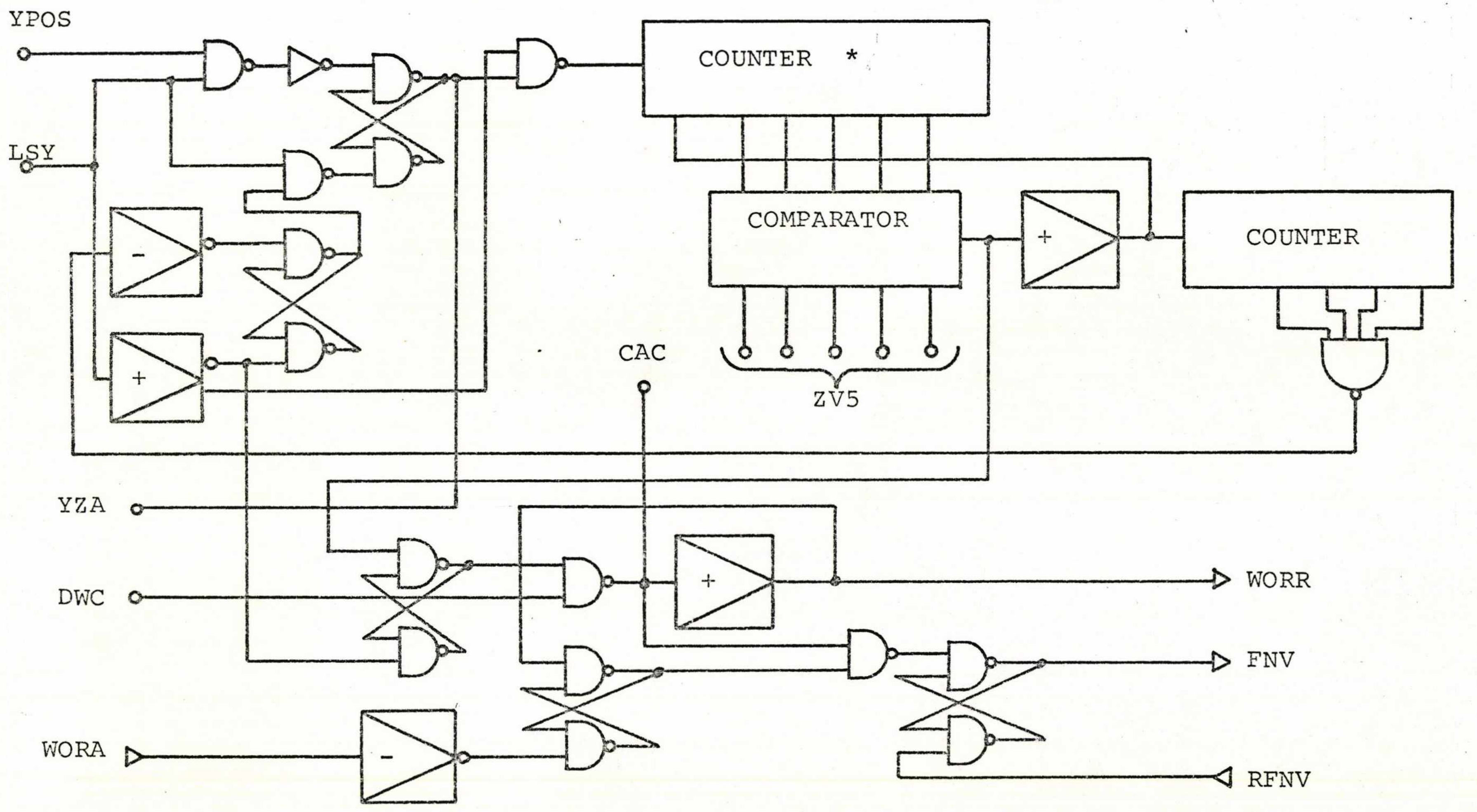
VIDEO PROCESSOR

FIGURE A1.6



X ZOOM COUNTER AND
LINE SAMPLER

FIGURE A1.7



Y ZOOM COUNTER

FIGURE A1.8

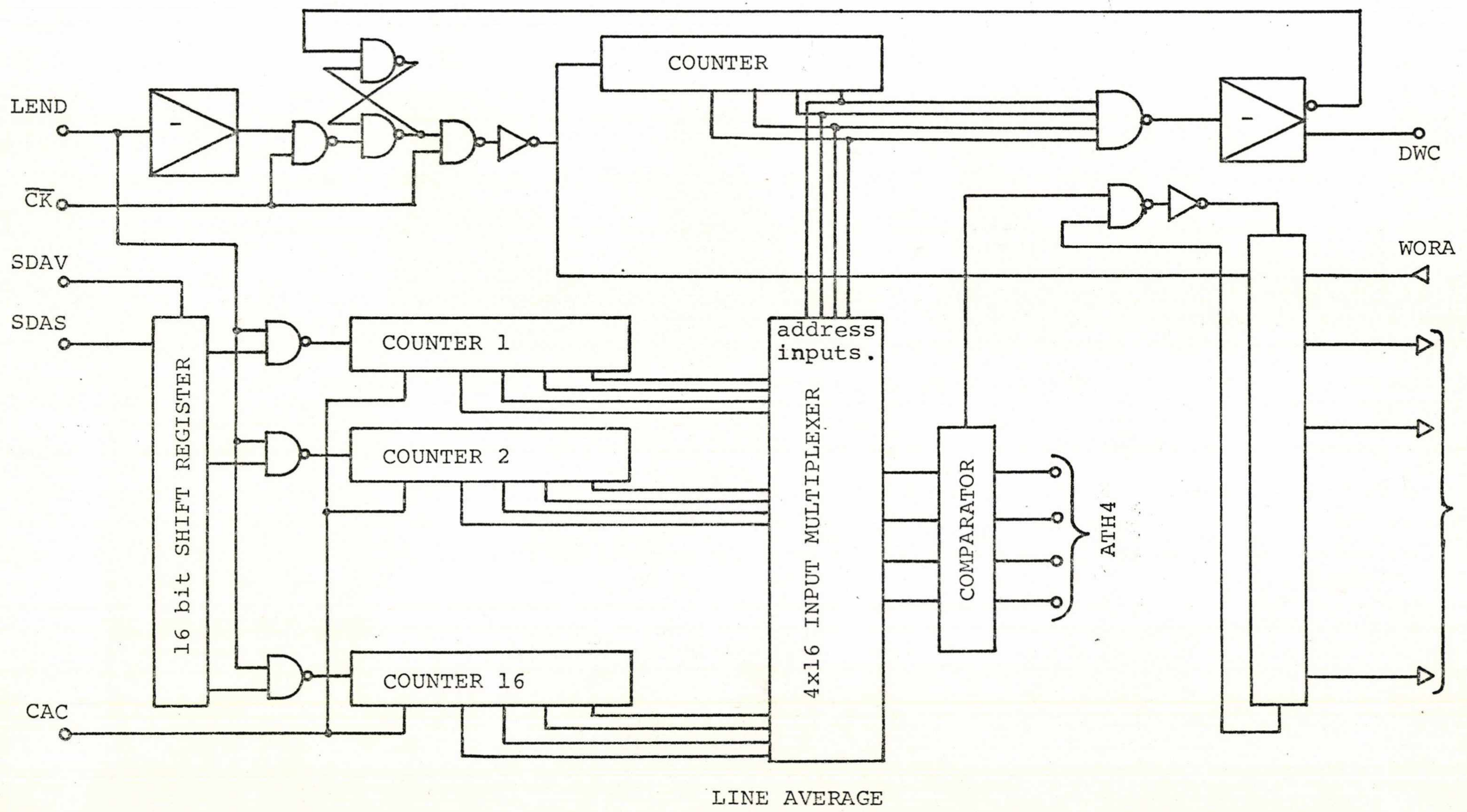
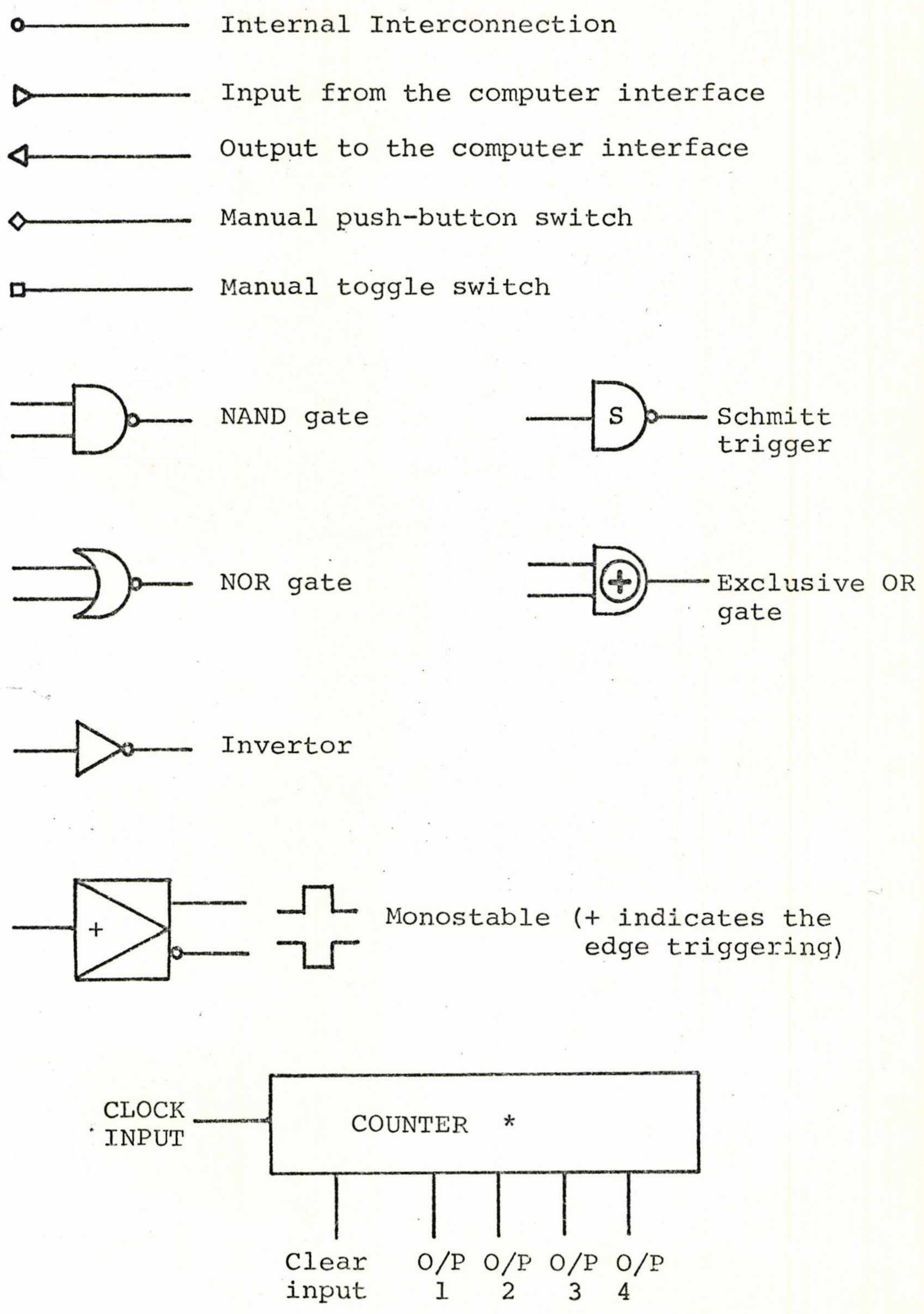
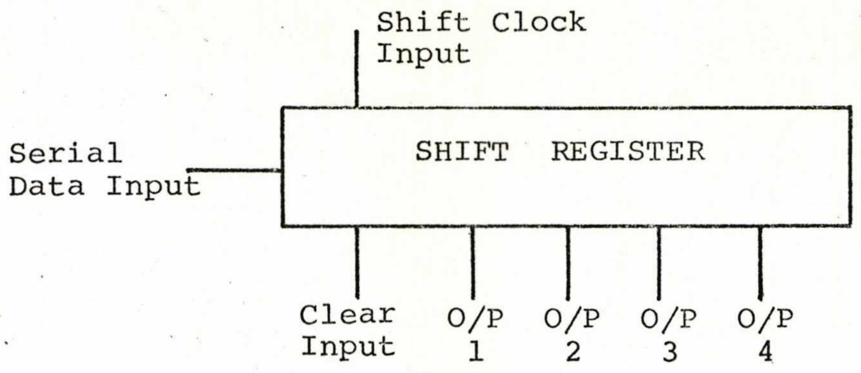

LEND
CK
SDAV
SDAS
CAC
16 bit SHIFT REGISTER
COUNTER
COUNTER 1
COUNTER 2
COUNTER 16
address inputs.
4x16 INPUT MULTIPLEXER
COMPARATOR
ATH4
DWC
WORA
OUTPUT DATA
LINE AVERAGE

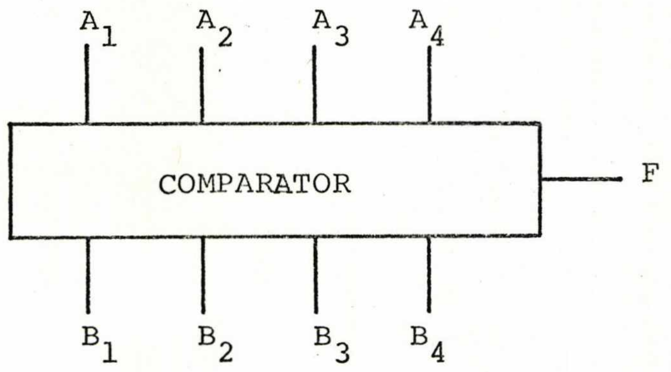
FIGURE A1.9



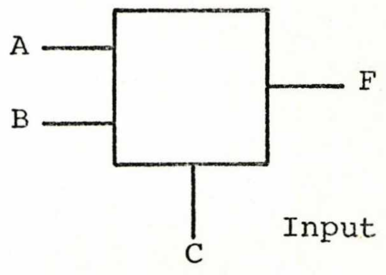
*signifies a +ve edge triggered synchronous type otherwise a -ve edge triggered ripple-through type.

FIGURE A1.10





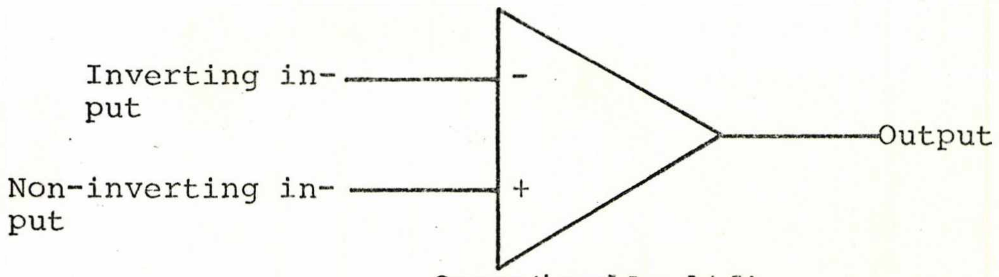
$F = \overline{\bigcap_{i=1,4} (A_i \oplus B_i)}$ (realised by 2 input Exclusive OR gates and a NAND gate)



$F = A.\bar{C} + B.C$

(realised by an AND-OR-Invert gate and an inverter)

Input Selector



Operational Amplifier

FIGURE A1.11

Appendix 2

Pattern Processing System
Subsystem Writers Manual

This manual is intended for users who wish to write an experiment subsystem for the Pattern Processing System and outlines the features which are available to him within the system.

There are two main command status within the system. The main program command status which allows manipulation of the data workspace and general functions and the executive command status which deals with assignment of peripheral channels etc. A full account of the functions available when in these command status are given elsewhere (Chapter 6).

On inputting the global command '*' the experiment subsystem will be entered. This subsystem may be any DAP program which is called EXPR by name, and the system causes a 'JST' to this label on entering the subsystem. An exit to the main program may be made by a 'JMP*' the entry point (EXPR).

The following description of the features of the system assume that the reader has a knowledge of DAP assembly language. In these descriptions, the following abbreviations will be used.

A	A register
B	B register
C	C register (1 bit)
D1	1st location after subroutine call, usually D1 is the return location from the subroutine
D2	2nd location after subroutine call
X	X register.

An argument of the form '(1-8)' after a register refers to the bits of that register which are currently relevent.

In this system the A, B and C registers are used for transferring parameters to subroutines. The X register value is always preserved. Unless otherwise stated, the return from a routine will be to D1.

The following routines are available to a user when writing a subsystem. Only a brief description of the functions are given, further details may be obtained from the program listings.

Peripheral Routines	NAME	FUNCTION
1.	TYPINA	Input 1 frame from command channel. Result in A(9-16).
2.	TYPOUT	Output 1 frame (A(9-16)) to command channel.
3.	RPT	Input 1 frame from data channel.
4.	PUNCH	Output 1 frame to data channel.
5.	VISTOP	Output 1 frame to the Vista.

cont. ...

	NAME	FUNCTION
6.	PRINTER	Output 1 frame to the line printer.
7.	DICHAN	Define input channel, A has device number.
8.	DOCHAN	Define output channel, A has device number.
9.	IDCHAN	Command input to defined channel.
10.	ODCHAN	Command output to defined channel.
11.	ONCHAN	Restore command output channel to default device.
12.	INCHAN	Restore command input channel to default device.
13.	RMCHAN	Remember the state of the channels.
14.	RSCHAN	Restore the channels to the last remembered state.
15.	SEQU	Set in tape control mode.
16.	NSEQ	Reset system to normal control mode.
17.	CANP	Cancel messages which are output on the command channel.
18.	NCAN	Reset the above function.

Camera Subroutines

19.	OPIC	Obtain frame from the camera using the settings in the main program.
20.	NEWX	Change value of X parameter by the value in A. If successful

<u>cont. ...</u>	<u>NAME</u>	<u>FUNCTION</u>
		on the return A=0 otherwise, A=-1.
21.	NEWY	As above, but for Y parameter.
22.	NEWZ	Change Zoom value by the value of A. On return A=0 if successful and -1 otherwise.
23.	NEWC	As above, except this sub-routine changes the Zoom with the center of the viewing window as a reference.
24.	GPIC	Obtain frame from the camera with the following settings D1=X value D2=Y value D3=Zoom value D4=brightness level D5=Special options D6 is the return location.

<u>Command channel character string inputting routines</u>

25.	INCOMMAND	A=Starting location of return pointers, D1 is the start of the list of mnemonics. This routine is for inputting and decoding one and two character mnemonics. When the user inputs a mnemonic it is compared with a list of mnemonics which follows the subroutine call and when the correct one is detected the routine returns by jumping through a corresponding location in a list of pointers. The list

cont. ...	NAME	FUNCTION
		of mnemonics is terminated by a location set to O. If the input mnemonic is not in the list of mnemonics, the routine returns to the first location after the end of the pointer list.
26.	INUMBER	Inputs a number, the result is in A.
27.	INCOMP	Inputs a number with limits. On entry A=maximum limit, B=minimum limit. The result is in A.
28.	YORN	For inputting Yes/No answers. If Y is input A=-1 on returning, and if N is input, A=O.
29.	YENO	For inputting Yes/No answers. If Y is input A=-1 and a return is made to D1. If N is input A=O and a return is made to D2. If any other character is input A(9-16) is set to that character and a return is made to D3.

Command channel outputting routines

3O.	MESSAGE	This outputs a 'message' (Character string) to the command channel. D1 contains a pointer to the character string and after outputting a return is made to D2.

cont. ...	NAME	FUNCTION
31.	DOUTNUMBER	Outputs the number in A in decimal with suppressed lead zeros.
32.	BOUTNUMBER	As above but in binary.
33.	OOUTNUMBER	As above but in octal.
34.	DOUTWORD	Outputs the number in A in decimal as a 16-bit integer with suppressed lead zeros.
35.	BOUTWORD	As above but in binary.
36.	OOUTWORD	As above but in octal.
37.	BOOUTWORD	Outputs the number in A in binary coded Octal format.
38	FDOUTNUMBER	Outputs the number in A in decimal with lead zeros.
39.	FOOUTNUMBER	As above but in octal.
40.	FBOUTWORD	Outputs the number in A as a 16-bit integer in binary with lead zeros.
41.	CRLF	Outputs a new line.
42.	OKE	Outputs 'O.K.' followed by a new line.

<u>Other character string routines</u>

There is a general purpose character buffer (128 characters long) which is used by the following routines:

43.	BUFO	Obtains the state of the buffer. On returning A=the starting location of the buffer and B=the number of 16-bit words which the buffer contains.

cont. ...	NAME	FUNCTION
44.	INBUF	The character in A(9-16) is inserted after the last character in the buffer. On returning A=-1 if this is not possible and 0 otherwise.
45.	OUTBUF	One character is taken from the start of the buffer and is in A(9-16) when the routine returns. If the buffer is empty, the routine returns with A=-1.
46.	INLINE	This enables the user to input a line of characters via the command channel. On return A(2-16) contains the number of characters input and if the line has been terminated by '#' A(1) is set.
47.	OUTLINE	This outputs the characters in the buffer to the command channel.
48.	OPRINT	This outputs the characters in the buffer to the line printer.

It is possible to output characters on paper tape, for headings, in visible format so that the user can read them. This can be done with the following routines:

cont. ...	NAME	FUNCTION
49.	VIZP	This outputs one character (A(9-16)) to the data channel in visible format.
50.	POVZ	This routine uses INLINE to input a string of characters and then outputs them to the data channel in visible. format.

The following routines are concerned with manipulating 16x16 binary patterns:

51.	SORI	Convert a data store pattern number to a pointer to the first location of that pattern. On entry A=pattern number on return A=required starting location. If this is not successful the return is made to D-2.
52.	DESORT	Convert a pattern pointer to a pattern number. On entry A=pointer and on returning A=pattern number. If this is not successful A=-1 on returning.

For the routines 53 to 63 the A register on entry contains a pointer to the starting location of the pattern to be operated on.

53.	CLEA	Set all locations of the pattern to 0.
54.	FILL	Set all locations of the pattern to 1.

cont. ...	NAME	FUNCTION
55.	NEG	Complement all locations of the patterns.
56.	REMX	Input the pattern from the data channel (paper tape format).
57.	OMX	Output the pattern to the data channel (paper tape format).
58.	OBP	Output the pattern to the command channel as 16 16-bit binary numbers.
59.	OBCO	Output the pattern to the command channel as 16 binary coded octal numbers.
60.	INS	Input the pattern from the command channel as 16 numbers.
61.	ALT	Replace a row of the pattern specified by a number input from the command channel by a second number input from the command channel.
62.	CROT	Rotate the pattern clockwise by 90°.
63.	AROT	Rotate the pattern anticlockwise by 90°.

For the routines 64 to 68 B on entry contains a pointer to the start of the pattern to be operated on and A contains a number N :

64.	MASK	Set the first N bits of the pattern to 1 and reset the remainder.
65.	TOPROL	Roll the pattern N rows toroidally.

cont. ...	NAME	FUNCTION
66.	SIDROL	Roll the pattern N columns toroidally.
67.	REFP	Find the value of an element of the pattern.A(1-12) specifies the Y coordinate and A(13-16) specifies the X coordinate. On returning A=0 if the element is reset and A=-1 if the element is set.
68.	REFT	Set the value of an element of the pattern. The value of this element is in C on entering and the location is specified in the same way as for 67.

For the routines 69 to 75 a pattern (P2), pointed at by the value in B,is operated on by the pattern (P1), pointed at by the value in A.

69.	MOVE	Transfer P1 to P2.
70.	INT	Interchange P1 and P2.
71.	AND	AND P1 and P2 the result is in P2.
72.	ORE	OR P1 and P2 the result is in P2.
73.	XOR	Exclusive OR P1 and P2.
74.	BITS	Find the Hamming distance between P1 and P2. The result on returning is in A.

The routines 75 to 80 require three or more parameters to be transferred when entered:

cont. ...	NAME	FUNCTION
75.	SHFTR	This routine treats a pattern as a 256-bit shift register. The new bits to be input are in A, the pointer to the pattern is in B and the number of shifts $n (0 \leqslant n \leqslant 16)$ is in D1. The routine returns to D2 with the bits that have overflowed from the register in A.
76.	RSHIFT	This behaves in similar way to SHIFTR except that the shift register is shifted in the opposite direction.
77.	IMP	This routine replaces bits in a pattern P2 by correspondingly located bits in a pattern P1 where there are 1's in a pattern P3. On entering this routine A points to P1, B to P2 and D1 to P3. The routine returns to D2.
78.	MAP	This routine is to map one pattern which is pointed to by D1 to a second pattern pointed to by A. The 256 8-bit list for directing the map is stored in 8 consecutive patterns which are pointed at by B. The routine returns to D2.

cont. ...	NAME	FUNCTION
79.	RMAP	This routine is organised in the same way as MAP except that it performs the inverse mapping function. Hence, for a 1 to 1 mapping the original pattern which produced a mapped pattern by MAP may be recreated by using RMAP on the mapped pattern.
80.	BMAP	This is a more general mapping routine than MAP in that it may map n consecutive patterns to n patterns where $n \leqslant 256$. To do this one word is used to specify the mapping of each point, hence, the connection list is 16n patterns long. This routine is organised in a similar way to MAP except that n is contained in D2 and the return is made to D3.

Miscellaneous Routines

81.	SERCH	This compares a word in A with a list of code words. This routine is organised in a similar way to INCOMMAND (25) except that it is entered with the unrecognised code in A and the pointer to the list of pointers in B.

cont. ...	NAME	FUNCTION
82.	ADUMP	This routine is a subsystem which enables the user to set parameters for the dumping routine PDUMP from the command channel.
83.	PDUMP	This outputs a consecutive block of patterns in character format. A contains the 1st pattern number of the block and B contains the last.
84.	RANDOM	This routine randomly sets n bits in a 16-bit word. The value of n is in A on entering and the desired random word is in A when the routine returns.
85.	SETSCAN	This is a subsystem to enable the user to set the parameters for the SCAN.
86.	TSCAN	This routine scans a scene with the camera and stores the result in a selected area of the data workspace.
87.	CSCAN	This routine scans a scene with the camera and compares the result in Hamming distance with a previously stored scan in the data workspace.

cont. ...	NAME	FUNCTION

The following routines (88 to 96) are concerned with controlling the Magnetic tape handler. The mag tape format for a 'file' within this system is defined as follows. A file number L followed by a character file of M lines followed by a data file of N patterns. The file numbers L are consecutive starting with 1 at the beginning of the tape. M or N for a file may be zero, the maximum limit is determined by the length of the tape. If an error occurs the routines return with -1 in A, otherwise they return with 0 in A.

88.	TWFILE	Opens a new file and writes a file number.
89.	TWCHAR	Writes a character line (from the general purpose buffer).
90.	TWPAT	Writes a block of consecutive patterns. The number of patterns to be written is in A and the first pattern number is in B.
91.	TRFILE	Reads a file number (the number is returned in B).
92.	TRCHAR	Reads a line of a character file and puts it into the general purpose buffer.
93.	TRPAT	Reads a block of n data patterns. The routine is entered with n in A and the first pattern number for the data to be stored in B.
94.	TEND	Moves the mag tape to the end of the last file.

cont. ...	NAME	FUNCTION
95.	TMOV	Moves the mag tape to the file specified in A.
96.	TBOT	Moves the mag tape to the beginning.

There are some Global locations which may be useful to the user. These are as follows:

1.	DSTO	This is the location of the first pattern of the data workspace.
2.	K816	This location contains the address of the top of the available data store.
3.	Location '63	Contains a pointer to the entry point of the executive.
4.	STRT	This location is the starting location of the main program.
5.	MINB	This location contains the starting address of the defined SLAM simulation store.
6.	MINE	This location contains the address of the last pattern which has been allocated for the SLAM simulation store.
7.	XPOS	This location contains the current X-coordinate of the camera viewing window which is used in the main program.
8.	YPOS	As above but the Y-coordinate.
9.	MAG	As XPOS but for the Zoom value.
10.	MG16	This is the value in MAG multiplied by 16.

INDEX OF GENERAL ROUTINES

<u>Learning Network Routines</u>

A library of routines is available which can operate the learning machine Minerva or simulate SLAM elements within the data workspace. The names and functions of these routines are as follows:

	NAME	FUNCTION
		Routines 1 to 12 are for operating Minerva.
1.	MUSCRE	Reset a card of four SLAM-16s. The address of the card is in A.
2.	MUSCSE	Set a word, the address of the card is in A.
3.	MUSCTO	Teach a card of SLAMs 0. The input pattern is in A and the address of the card is in B.
4.	MUSCT1	As above but teach 1.
5.	MUSCOP	Obtain an output from a card. The input pattern is in A, the card address is in B and the 4-bit result is put into A(13-16).
6.	MUSPRE	Reset a 'pattern' of Minerva SLAMs 16 consecutive cards are referred to as a pattern of SLAMs because they sample a 16x16 bit input pattern. The starting address is in A.
7.	MUSPSE	As above but set instead of reset.
8.	MUSPTO	Teach 1 to a pattern of Minerva cards. The starting location of the input pattern in store is in A and the address of the first Minerva card is in B.

cont. ...	NAME	FUNCTION
9.	MUSPT1	As above, but teach 1 instead of 0.
10.	MUSPOP	Obtain the total number of 1's output from a pattern of Minerva cards. On entry, the registers are set up as described for MUSPTO. On returning the summed response is in A.
11.	MUSPPO	Probablisticly teach D1 to a pattern of Minerva cards. The registers are set up as described for MUSPTO. D1 contains the 16-bit probablistic teach vector and the routine returns to D2.
12.	MUSPP1	As above, but teach 1 instead of 0.

The following routines 13 to 43 are concerned with simulating SLAMs in the data workspace. 1 16-bit word is used to simulate each SLAM 16.

13.	MINSET	This enables a user to select an area of data workspace for the SLAMs to be simulated. The SLAM routines may at run time only access this area for simulating SLAMs.

Routines to simulate 1 SLAM 16 (14-18).

14.	SALR	Reset a SLAM the location of which is in A.
15.	SALS	Set a SLAM the location of which is in A.
16.	SALD	Teach a SLAM d, the input pattern is in A(13-16) and the location of the SLAM is in B.

cont. ...	NAME	FUNCTION
17.	SAL1	As above, but teach 1.
18.	SALOP	Obtain an output from a SLAM. The registers on entry are the same as for SALO. If the output is 1 then A=-1, if the output is O then A=O on returning.

Routines to simulate a Minerva card of SLAM 16s.

19.	MINR	Reset a card of SLAMs the starting location is in A.
20.	MINS	Set a card of SLAMs the starting location is in A.
21.	MIND	Teach a card of SLAMs O the input pattern is in A and the starting location of the SLAMs is in B.
22.	MIN1	As above, but teach 1.
23.	MINOP	Obtain the output from a card of SLAMs. The registers are set up in the same way as for MINO. On returning the result is in A(13-16).

Routines to simulate a 'pattern' of SLAMs which consists of 16 consecutive cards of SLAMs.

24.	MINPR	Reset a pattern of SLAMs. The starting location of the SLAMs is in A.
25.	MINPS	As above, but set instead of reset.
26.	MINPO	Teach O to a pattern of SLAMs. The starting location of the input pattern is in A and the starting location of the SLAMs is in B.

cont. ...	NAME	FUNCTION
27.	MINP1	As above, but teach 1.
28.	MINPOP	Obtain the sum of all the outputs of a pattern of SLAMs which are 1. The registers are set up in the same way as for MINPO and on returning the summed response is in A.
29.	MINPPO	Probablisticly teach 0 to a pattern of SLAMs. The registers are set up as for MINPO, the 16-bit teach vector is in D1 and the routine returns to D2.
30.	MINPP1	As above, but teach 1.

The following routines are for simulating a card of 4 SLAMs without the restrictions on the teach inputs which occurs in Minerva. A data word is associated with the card, which is formed in the following way. Bits (1-4) signify the teach sense values, bits (5-8) signifies the teach clock values and bits (13-16) contain the last obtained output from the card.

31.	MISLCT	Teach a card of SLAMs. The input pattern is in A. The starting location of the SLAMs is in B and the data word is in D1 and the routine returns to D2. (All the other card functions may be conducted by the normal card routines (19, 20 and 23)).

cont. ...	NAME	FUNCTION

The following routines are for simulating a 'pattern' of the cards of SLAMs mentioned above. With each 'pattern' of these SLAMs a data pattern of 16 consecutive data words is associated.

32.	MISLPT	Teach a pattern of SLAMs. A has the location of the input pattern, B the location of the SLAMs and D1 the location of the data pattern. The routine returns to D2.
33.	MISLPO	Obtain a new output from a pattern of SLAMs. The routine is entered as described above for MISLPT. (All other pattern functions may be conducted by the normal pattern routines (24, 25 and 27).

The following routines (34 to 54) are concerned with the simulation of cumulative learning SLAMs (CL SLAMs). Routines for 4 output CL SLAM 16s: the simulation of 4 output CL SLAM 16 requires 4 16-bit words.

34.	SALCLC	Clear a CL SLAM (i.e., set all the outputs to 8). The location of the CL SLAM is in A.
35.	SALCLT	Teach on output to a CL SLAM. The input pattern is in A(13-16) and the teach number is in A(5-8). The location of the SLAM is in B.

cont. ...	NAME	FUNCTION
36.	SALCLO	Obtain the output from a CL SLAM. The input pattern is in A(13-16). The location of the SLAM is in B and the output is in A when the routine returns.

Routines to simulate a card of 4 4-output CL SLAM 16s.

37.	MINCLC	Clear a card of CL SLAMs the location of the first SLAM is in A.
38.	MINCLT	Teach each CL SLAM an output pattern. The input pattern is in A, the location of the first SLAM is in B and the output patterns are in D1(5-8) and (13-16) and D2(5-8) and (13-16). The routine returns to D3.
39.	MINCLI	Increment the output of a card of CL SLAMs. The input pattern is in A, the location of the first SLAM is in B and the teach sense is in C. If C=1, then the SLAMs are incremented; if C=0, then they are deincremented. On returning, the number of SLAMs which have saturated is in A.
40.	MINCLO	Obtain the outputs from a card of CL SLAMs. The input pattern is in A and the location of the first SLAM is in B. On returning, the four responses are in A(5-8), A(13-16), B(5-8) and B(5-16).

cont. ...	NAME	FUNCTION
Routines to simulate a pattern of CL SLAMs.		
41.	MINPCC	Clear a pattern of CL SLAMs. The location of the first SLAM is in A.
42.	MINPCI	Increment the output of a pattern of CL SLAMs. The location of the input pattern is in A, the location of the first SLAM is in B and the teach sense is in C. On returning, the total number of CL SLAMs which have saturated is in A.
43.	MINPCO	Obtain a summed response from a pattern of CL SLAMs. The location of input pattern is in A, the location of the first SLAM is in B and on returning, the summed response is in A.

The following routines are concerned with 16 output CL SLAM 16s. For programming convenience, one word represents one element of the SLAM and a 16 output CL SLAM 16 is simulated by 16 consecutive words.

Routines for one 16 output CL SLAM 16.

44.	BISLC	The same function as SALCLC (34) except the mid value is 32,568 for a 16 output CL SLAM.
45.	BISLI	The same function as SALCLT (35) except that the teach number is in D1(1-16) and the routine returns to D2.

cont. ...	NAME	FUNCTION
46.	BISLO	The same function as SALCLO (36).

Routines for a card of four 16 output CL SLAM 16s.

47.	BISLCC	The same function as MINCLO (37).
48.	BISLCI	The same function as MINCLI (38).
49.	BISLCO	A similar function to MINCLO (40) except that the 4 outputs are summed and the routine returns with the response as a double precision integer in A and B.

Routines for a pattern of 16 output CL SLAM 16s.

50.	BISLPC	The same function as MINPCC (41).
51.	BISLPI	The same function as MINPCI (42).
52.	BISLPO	The same function as MINPCO (43) except that the response is a double precision integer in A and B.

The following, output incrementing, teach routine has two additional features. Firstly, it can probabilisticly teach a pattern of CL SLAMs. Secondly, the size of the increment may be varied hence n output CL SLAMs may be simulated where $1 \leqslant n \leqslant 16$.

53.	BIGPI	Similar to MINPI (27) except that D1 contains the 16-bit teach vector and D2 contains the increment size. The routine returns to D3. (All other functions can be conducted by the normal routines 51 and 53).

cont. ... NAME FUNCTION

The following routines are for simulating a card of 4 TR SLAM 16s. This is achieved by using two cards of normal SLAMs, one for the data output and one for the valid output.

cont. ...	NAME	FUNCTION
54.	MITRCT	This is similar to MISLCT (31) except that bits (9-12) of the data word are used for the last valid output.
55.	MITRCO	This obtains an output from a card of TR SLAMs. It is entered in a similar way as MISLCT (31). On returning, the complete data word is in A with bits (9-12) containing the new valid output and bits (13-16) containing the new data output. (A card of TR SLAMs may be reset by using MINR twice).

References

1. ALEKSANDER, I. 'Microcircuit Learning Computers', Mills & Boon, 1971.

2. ARBIB, M.A. 'Transformation and Somatotopy in Perceiving Systems', Second Int. Joint Conf. on Artificial Intelligence, PP.140-148, 1971.

3. BALL, A.G. 'Alan's Debug' : User Manual, Electronics, University of Kent, at Canterbury, 1971.

4. BALL, A.G. 'AMOS Sub-System Writers Manual' Electronics, University of Kent at Canterbury, February, 1973.

5. BLEDSOE, W.W. & BROWNING, I. 'Pattern Recognition and Reading by Machine', Proc. of the Eastern Joint Computer Conf. PP.225-232, 1959.

6. BLEDSOE, W.W. & BISSON, C.L. 'Improved Memory Matrices for the n-tuple Pattern Recognition Method', IRE Trans. on Electronic Computers, Vol.EC-11, No.3, PP.414-415, June, 1962.

7. CHEUNG, C.Y. 'Some Aspects of Adaptive Logic for Pattern Recognition', Ph.D. Thesis in Electronics, University of Kent at Canterbury, 1973.

8. CHUNG, S.H. 'Neurophysiology of the Visual System' in 'Recognising Patterns', Eds.: Kolers, P.A. & Eden, M. The M.I.T. Press, PP.82-101, 1968.

9. DEUTSCH, E.S. 'Character Preprocessing and Recognition : A Pseudo-Topological Approach', Ph.D. Thesis in Engineering, University of London, 1969.

10. DINN, D.F. WINTER, D.A. & TRENHOLM, B.G. 'CINTEL-Computer Interface for Television', IEEE Trans. on Computers, Vol.C19, PP.1091-1095, November, 1970.

11. DUDA, R.O. & HART, P.E. 'Pattern Classification and Scene Analysis', John Wiley & Sons, 1973.

12. EDEN, M. 'Handwriting Generation and Recognition' in 'Recognising Patterns' Eds.: Kolers, P.A. & Eden, M. The M.I.T. Press, PP.138-154, 1968.

13. FAIRHURST, M.C. & ALEKSANDER, I. 'Natural Pattern Clustering in Digital Learning Nets', Electronics Letters, Vol.7, No.24, PP.724-726, December, 1971.

14. FAIRHURST, M.C. 'The Dynamics of Learning in Some Digital Networks', Ph.D. Thesis in Electronics, University of Kent at Canterbury, 1973.

15. FORSEN, G.E. 'Processing Visual Data With An Automaton Eye' in 'Practical Pattern Recognition', Thompson Book Company, PP.471-502, 1968.

16. FUKUSHIMA, K. — 'Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements', IEEE Trans. on System Science and Cybernetics, Vol.SSC-5, No.4, PP.322-333, October, 1969.

17. GLOVER, R.J. ALEKSANDER, I. & REEVES, A.P. — 'The Minerva Adaptive Computing System' (To be published).

18. GOLAY, M.J.E. — 'Hexagonal Parallel Pattern Transformations', IEEE Trans. on Electronic Computers, Vol.C-18, PP.733-740, August, 1969.

19. GRIMSDALE, R.L. SUMNER, F.H. TUNIS, C.J. & KILBURN, T. — 'A System for the Automatic Recognition of Patterns', Proceedings of the IEE., Vol.106, Part B, No.26, PP.210-221, March, 1959. Also, in 'Pattern Recognition', Uhr, L. Editor. John Wiley & Sons, PP.317-338, 1966.

20. HEBB, D.O. — 'The Organisation of Behaviour', John Wiley & Sons, 1949.

21. HERSCHER, M.B. & KELLY, T.P. — 'Functional Electronic Model for The Frog Retina', IEEE Trans. on Military Electronics, Vol.7, PP.98-103, April-July, 1963.

22. HOSKING, K.H. & THOMPSON, J. — 'A Feature Detection Method for Optical Character Recognition', Conf. on Pattern Recognition, Organised by IEE & NPL, PP.271-283, 1968.

23. HOSKING, K.H. 'A Contour Method for the Recognition of Hand Printed Characters', Machine Perception of Patterns and Pictures, Conf. series no.13, The Instit. of Physics, London & Bristol, PP.19-27, 1972.

24. HUBEL, D.H. & WIESEL, T.N. 'Receptive Fields, Binocular Interaction and Functional Architecture in the Cats Visual Cortex', Journal of Physiology, Vol.160, PP.106-123, 1962. Also, in 'Pattern Recognition', Uhr. L. Editor. John Wiley & Sons, PP.262-277, 1966.

25. HUNT, D.J. 'A Feature Extraction Method for the Recognition of Hand Printed Characters', Machine Perception of Patterns and Pictures, Conf. series no.13, The Instit. of Physics, London & Bristol, PP.28-33, 1972.

26. KABRISKY, M. 'A Proposed Model for Visual Information Processing in the Brain', University of Illinois Press, 1966.

27. KELLEY, M.D. 'Edge Detection in Pictures by Computer Using Planning', in 'Machine Intelligence 6', Meltzer, B. & Miche, D. Eds. PP.397-404, 1971.

28. KOLERS, P.A. 'Some Psychological Aspects of Pattern Recognition' in 'Recognising Patterns', Kolers, P.A. & Eden, M. Eds. The M.I.T. Press, PP4-61, 1968.

29. LEDLEY, R.S.	'High-Speed Automatic Analysis of Biomedical Pictures', Science, 146, PP.216-223, October, 1964.
30. LEGÉNDY, C.R.	'How Large are Hebbs Cell Assemblies?', in 'Cybernetic Problems in Bionomics', Bionomics Symposium, 1966. Oestricher, H.L. & Moore, D.R. Eds. PP.721-724.
31. LETTVIN, J.Y. MATURANA, H.R. McCULLOCK, W.S. & PITTS, W.H.	'What the Frog's Eye Tells the Frog's Brain', Proc. IRE., Vol.47, PP.1940-1951, 1959.
32. MASON, S.J. & CLEMENS, J.K.	'Character Recognition in an Experimental Reading Machine for the Blind', in 'Recognising Patterns', Kolers, P.A. & Eden, M. Eds. The M.I.T. Press, PP.156-167, 1968.
33. MINSKY, M. & PAPERT, S.	'Perceptrons', M.I.T. Press, 1969.
34. NOTON, D.	'A Theory of Visual Pattern Perception', IEEE Trans. on System Science and Cybernetics, Vol.SSC-6, No.4, PP.349-357, October, 1970.
35. NOTON, D. & STARK, L.	'Eye Movements and Visual Perception', Scientific American, PP.34-43, June, 1971
36. PARKS, J.R.	'A Multi-Level System of Analysis for Mixedfont and Hand-Blocked Printed Characters Recognition', in 'Automatic Interpretation and Classification of Images', edited by Grasselli, A. Academic Press, PP.295-322, 1969.

37. PINGLE, K.K. 'Visual Perception by a Computer', in 'Automatic Interpretation and Classification of Images', edited by Grasselli, A. Academic Press, PP.277-284, 1969.

38. REEVES, A.P. 'Visual Display Unit Report', Third Year Project Report, Electronics, University of Kent at Canterbury, 1970.

39. REEVES, A.P. 'Improvements In Adaptive Networks', U.K. Patent Applic. No.36006/71, 1971.

40. ROSENBERG, B. & WILKINS, B.R. 'Coding in the Visual System', Conf. on Pattern Recognition. Organised by IEE & NPL, PP.77-85, 1968.

41. ROSENBLATT, F. 'Principles of Neurodynamics : Perceptrons and the Theory of Brain Mechanisms', Spartan Books, 1962.

42. RUNGE, R.G. UEMURA, M. & VIGLIONE, S.S. 'Electronic Synthesis of the Neural Network in the Pidgeon Retina', in 'Cybernetics Problems in Bionomics', Bionomics Syposium, 1966. Oestricher, H.L. & Moore, D.R. Eds. PP.791-810.

43. SARAGA, P. WEAVER, J.A. & WOOLLONS, D.J. 'Optical Character Recognition', Philips Technical Review, Vol.28, PP.197-203, 1967.

44. SARAGA, P. & WAVISH, P.R. 'Edge-coding Operators for Pattern Recognition', Electronics Letters, Vol.7, No.25, PP.736-738, December, 1971.

45. SARAGA, P. & WOOLLONS, D.J.	'The Design of Operators for Pattern Processing', Conf. on Pattern Recognition. Organised by IEE & NPL, PP.106-116, 1968.
46. SCAN-DATA CORP.	'Scan Data Optical Character Reading System', Reference Manual.
47. STECK, G.P.	'Stochastic Model for the Browning Bledsoe Pattern Recognition Scheme', IRE Trans. on Electronic Computers, Vol.EC11, PP.274-282, April, 1962.
48. SUTRO, L.L.	'Proposed Electronics to Represent Properties of the Frog's Eye' in 'Cybernetic Problems in Bionomics', Bionomics Symposium, 1966. Oestricher, H.L. & Moore, D.R. Eds. PP.811-819.
49. SYMONS, M.	'A New Self-Organising Pattern Recognition System', Conf. on Pattern Recognition. Organised by IEE and NPL, PP.11-20, 1968.
50. TAYLOR, W.K.	'Learning Characteristics of a Trainable Pattern Recognition Machine', Conf. on Pattern Recognition. Organised by IEE & NPL, PP.238-249,1968.
51. TOLLYFIELD, A.J.	'Investigation of the Behaviour of SLAM Nets with Feedback', First Year Report, Electronics, The University of Kent at Canterbury, 1971.

52. TOUSSANT, G.T. & DONALDSON, R.W. 'Algorithms for Recognising Contour-Traced Hand-Printed Characters', IEEE Trans on Computers, Vol.C19, PP.541-546, June, 1970.

53. UHR, L. & VOSSLER, C. 'A Pattern Recognition Program that Generates, Evaluates & Adjusts Its Own Operators', Proc. of the Western Joint Computer Conf. Vol.19, PP.555-569, May, 1961. Also, in 'Pattern Recognition', Uhr, L. Editor. John Wiley & Sons, 1966.

54. ULLMANN, J.R. 'Experiments with the n-tuple Method of Pattern Recognition', IEEE Trans. on Computers, Vol.C18, PP.1135-1137, December, 1969.

55. WATT, A.H. & BEURLE, R.L. 'Recognition of Hand-Printed Numerals Reduced to Graph Representable Form', Second Int. Joint Conf. on Artificial Intelligence, PP.322-332, 1971.

56. YARBUS, A.L. 'Eye Movement and Vision', Translation editor Riggs, L.A. Plenum Press, N.Y., 1967.

57. ZUSNE, L. 'Visual Perception of Form', Academic Press, 1970.