THE DESIGN AND APPLICATION OF ASSOCIATIVE MEMORIES FOR SCENE ANALYSIS

by

James Austin

Submitted For PhD.

Department of Electrical Engineering

Brunel University

August 1986

BRUNEL UNIVERSITY, UXBRIDGE.

DEPARTMENT OF ELECTRICAL ENGINEERING.

Author: James Austin

Title: The Design and Application of Associative Memories for Scene Analysis.

1986

ABSTRACT

This thesis investigates a novel scene analysis system that determines the identity and the relative positions of unconstrained objects within a natural three dimensional grey scale image. Images may be of `block filled` or `line drawn` occluded shapes. It utilises the occluding information to discover the relative depth of objects in the scene. The system incorporates associative memories, the N tuple pattern recognition process, movable multiple resolution windows and edge detection. The structure and performance of the system and its subsystems is reported. The associative memory incorporates a novel recall procedure which has uses outside the application given here. The work incorporates ideas from the neurophysiology of the human visual system to overcome some of the problems encountered.

Acknowledgments

I would like to thank Prof. Igor Aleksander for the initial project specification and supervision for the first two years of the work and Dr John Stonham for supervising the PhD in the latter part of the work. I would also like to thank the SERC for the funding. Finally I thank my wife Jayne for her patience and perseverance in checking through the initial draft of the thesis.

CHAPTER INDEX

FIGURE INDEX

APPENDIX INDEX

TABLES OF RESULTS

CHAPTER 1

Introduction

1. Introduction

The research reported in this thesis addresses the subject of scene analysis. The particular area of study is the recognition of occluded objects in two and three dimensional images. The aim of the work is the investigation of a scene analysis system which is capable of recognising objects in a ´natural´ and unconstrained environment, and report the relative three dimensional positions of any objects identified. Emphasis is placed upon the use of occluding information to assess the relative depth of objects within the field of view. Furthermore, the system is designed to learn the descriptions of objects by example, in that objects that are to be recognised need only to be placed in the visual field of the system and the system set to a learn mode.

The desired attributes of the system are as follows.

The identification of objects present within a scene. That is, the assignment of a classification label to every object in a scene.

The analysis of the scene such that the relative position of all objects are given.

Recognition of objects which may be ´line drawn´- i.e pencil and paper sketches, or which may be ´block filled´- i.e present within a ´natural´ scene.

Objects may be recognised in an unconstrained environment. i.e the illumination of the object need not be highly constrained to allow

successful recognition. Objects may be placed anywhere in the field of view of the system, recognition of rotated and scaled objects is possible by prior teaching.

Object descriptions are learned by examples.

The approach taken to the work is primarily at a theoretical level, in that the system is not constrained to one particular application. The primary objective is an investigation of the use of parallel processing systems that are simple models and approximations to systems which are thought to exist in the human nervous system. This has involved the use of many ideas from neurobiology and psychology in the design of the system, such as distributive memory, edge detection and foveal processing. The result has been a new approach to the problems found in scene analysis.

The system has been designed to be readily constructed in hardware, in that emphasis has been placed upon the computational restrictions this consideration imposes (cost and speed).

The work has resulted in three main innovations. These are, the general approach to occlusion analysis, the design of an affective content addressable memory, and a novel form of N tuple pattern recognition for grey scale image data.

As a result the work is of use in two subject areas. Firstly the problems of brain sciences - how information may be processed and stored in the brain, and secondly the design of a practical vision system.

The hardware concepts draw upon the N tuple pattern recognition technique. This has been applied successfully to the recognition of hand and type written data and the recognition of parts on a production line. The process has been implemented in WISARD, a dedicated pattern

recognition system.

This thesis may be broken up into five main sections, each covered by a number of chapters.

Chapters 2 and 3.

A review of scene analysis techniques and the N tuple pattern recognition process.

Chapter 4

An overview of the scene analysis technique to be used to discover occluding information.

Chapters 5,6,7 and 8

A review of associative memories, followed by the design of an associative memory suitable for the system proposed in chapter 4. Followed by an analysis of the storage properties of the memory.

Chapters 9,10,11 and 12

The design of the input processing stages of the recognition system, followed by experiments examining the abilities of the system.

Chapter 13

The internal description of the scene used by the system and its construction.

This is followed by a concluding chapter which identifies further work in the light of what has been achieved.

CHAPTER 2

Scene Analysis - A Review

2. Introduction

The object of this chapter is to review the approaches to scene analysis that are of relevance to this thesis, and to highlight the general problems of each approach and their specific problems in relation to occlusion analysis. Finally, the essential elements of the ideal scene analysis system will be identified.

Previous approaches will be discussed under 3 basic headings. These headings are somewhat arbitrary but are convenient in capturing the central features of each approach. First there are computational approaches which cover work considered to be in the realms of Artificial Intelligence. Next are pattern recognition approaches - forerunners of the AI approach, only more mathematical (less symbolic) and essentially `serial`. Finally, there are the parallel approaches - systems which have been specifically developed to be implemented on parallel hardware. The approach taken in this thesis comes under the last heading.

2.1. Computational Approaches

These approaches have been made by the AI community and are characterised by the use of a `structural description` of objects to be recognised. A `structural description` is a high level description of a shape, usually containing information on the vertices that make up an object and their relation to each other. The approach was restricted initially to scenes containing only polyhedra (see (Boden1977,Brady1981)) but later extended to more natural scenes not

just containing objects with straight edges. To begin with the work concentrated on extracting 3-dimensional descriptions from 2-dimensional scenes (Guzman1968) using geometric constraints applicable to 2D views of polyhedra only. The basic idea can be shown by a simple example. Consider the cube shown in Fig 2.1.

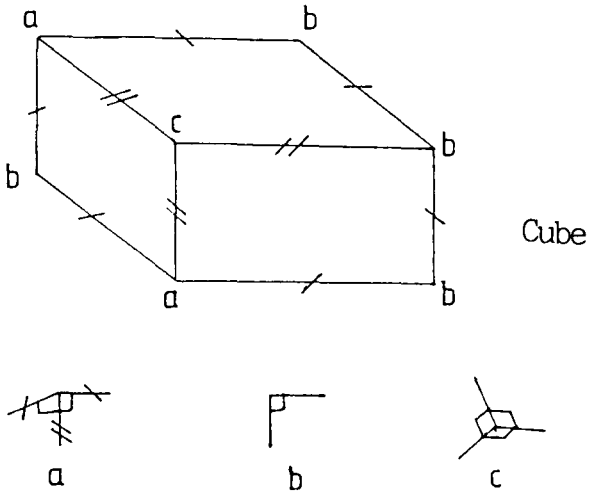


Fig 2.1

In considering how cubes are made up, it can be seen that only certain 3D constructions of vertices are possible, i.e those shown in Fig 2.1. Each vertex has a set of possible 3D interpretations, i.e vertex `a` in Fig 2.1 can be seen as a corner coming out at you, or a corner going in to the page (convex or concave). An initial assessment might give both as possible. Similar conclusions can be made about other vertices.

By taking one possible interpretation and applying the rule that an edge remains the same `type` along its length in polyhedral scenes (The type of edge is determined by the vertex and can be convex or concave) a restriction is placed on what form other vertices can take. As a result the image is reduced to a 3D description. The `cube` in

Fig 2.1 can be seen in two ways; as a cube or as a square with 2 sides added.

Note that no descriptions of the objects are needed to `see` the scene in 3D. The use of structural descriptions overcomes the ambiguity as in the above example of a cube (i.e. is it a cube or a square with two sides ?). If only one interpretation is known to be possible, then the scene is constrained to it. Also local searching can be applied when expected edges are not clearly defined, through noise etc. Such examples of using knowledge of objects in this `top down` manner is found in Shirai (Shirai1973).

An important aspect of the structural description of objects is their position, size and rotation invariance. A description of an object can be made by indicating the vertices, edges and edge types relating to the shape. This can then be matched to the data derived from the image. With this knowledge it would be possible to detect occlusion after the object has been matched with its description by finding which areas of the object are missing, and discovering if any objects are present in these areas.

Although the general method of occlusion analysis indicated above is used in the system developed in this thesis, the process of object identification was thought to be too slow and complex and not easily applied to `natural` scenes containing high amounts of noise. A further problem was in the acquisition of object descriptions; no fast way of deriving a stable description could be made, other than coding the three dimensional descriptions by hand. It can be noted here that human intervention is one indication of the complexity of problem, the apparently simple task of converting the image to a description that could be used by a machine could not be done by machine.

Another approach to scene analysis has been pursued by Marr (Marr1982) and is perhaps a cornerstone in vision research. It typifies the general approach made by a large part of the AI community. Most of his work has been aimed at making explicit basic parameters present within an image so that higher level systems are able to use the information in the process of recognition. The compilation of what is termed the 2 1/2 D sketch is done by extracting information present locally over the image, such as brightness, presence of surface discontinuations (edges), depth of points, followed by analysis of this to indicate motion and surface orientation of every point in the scene. The final `sketch` is a (very large) set of parameters describing the raw image. This process of setting parameters has been well documented, but the process of recognition has not been so well defined.

The following processes are needed before the raw data given by the above can be matched to some type of object description. Firstly a co-ordinate system needs to be imposed on the shapes present to indicate the relationship between these shapes. If one considers an image of a man , this process would define the principal axis, then go on to define axis of arms , hands, fingers etc. and the relationship between them. This description is then matched to the models in the database.

Little work has been done on the `recognition` stages of the theory and so conclusions about its abilities cannot be made. What is clear is the amount of processing involved , mostly at the low level, in the computation of the 2 1/2 D sketch. Convolutions with complex operators are needed, prompting the development of parallel arrays of multi-processor computers to enable processing in real time.

2.2. Pattern Recognition Approachs

The main application of the pattern recognition approach is not to `understand` the scene or put it in to context, i.e identify a man taking a dog for a walk, but concerns the problem of classification of shapes or, more generally `patterns` i.e. identifying a man and a dog within the scene. In some cases of scene analysis it may be unnecessary to actually classify objects - only to segment the scene into regions belonging to different objects. To do this motion information could be used. This point is made to clarify what pattern recognition is, i.e. the specific identification of patterns and assignment of a label to each object in an image.

The methods of pattern recognition are reviewed below giving the strengths and weaknesses of each. All the methods described in this section are serial processes, i.e. they are designed to be implemented on a serial computer with von Newman architecture. The obvious limitation of these serial processes is their speed of operation. In large scenes many operations are needed and, unless they are processed on multiple processor machines (i.e. parallel processing of serial algorithms), they are often extremely slow. If implemented on parallel processing machines the cost factor becomes important. More specific problems are covered under the descriptions of the individual methods.

2.2.1. Syntactic Pattern Recognition

Syntactic pattern recognition is based upon the use of grammars to express what an object is. It can be likened to the recognition of words in a sentence. We know that for the word `hello` the letters need to be arranged in a particular order, and if their order is different, a different word is present. Similarly if we consider a

shape, where the vertices are the letters and the edges indicate the order, a grammer can be used to discover what the shape is. For the shape shown in Fig 2.2 the sentence `abcd` can be used to characterise the object. Each letter characterises a vertex at a particular orientation, their order of presentation indicating how they are connected to form a shape. This string is named a `grammar` and is used for the recognition of the shape (Gonzalez1978) ,a number of such grammars for the recognition of many shapes being termed a `language`. Recognition is a simple process. The scene is converted into a string, and then parsed using the language constructed previously. The classification of the shape is given by the grammar that fits the present string.

This approach has been used by Ullman (Ullmann1983) on occluded images to decide which objects are present. His application has only been applied to 1D images. Such images are made up of strings of letters representing objects e.g. if two objects were represented `ABCDEF´ and `HIJKL´ , a possible 1D image containing these objects would be `ZZZZHIJABCDEFZZZZ` , where one object overlaps the other on a background of Z´s. The process is to be applied to 2D images in the future.

The main criticism of the syntactic approach is the problem of reducing the image to a string ; processes such as thining, edge detection and edge tracing have been applied to enable this. Given a noisy and/or incomplete object, the process of deriving the string describ-

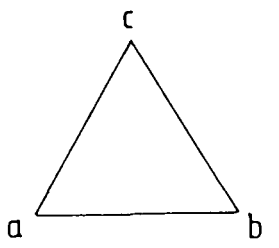


Fig 2.2
Simple object.

ing the scene and matching this to the grammars can be complex if not impossible - especially in scenes containing many objects. A further disadvantage is the difficulty in performing the process in parallel, enabling high processing speeds for large images. Since the operation is essentially serial only limited parallelism can be applied, mainly at the preprocessing stages.

Furthermore, selection of operators which identify the components making up the grammar (i.e such as lines, vertices, etc.) is difficult. The approach uses special operators which need to be programmed. Their reliability in detecting the features they are designed to recognise can only be guaranteed in a restricted environment, although the use of adaptable operators, as described in Aleksander and Wilsons paper (Aleksander1985) and also in chapter 3, may overcome this problem.

The strength of the syntactic process is in its ability to analyse scene information, i.e it is simple to define a string that describes an arch made of blocks (bottom left block, long block on to, bottom right block). The representation of an `arch` would be fairly position independent and could be recognised by applying the grammar for an arch to a low level set of descriptions of blocks and their positions. This use of syntactic pattern recognition at a higher level would also be fairly straightforward.

2.2.2. Discriminant Analysis

This process is based upon the idea of representing the patterns to be recognised in feature space, and then separating this space in such a way as to allow separation of the classes of object present. The idea of a feature space is shown in Fig 2.3. Each shape is plotted in feature space depending on the number of vertices and its average grey

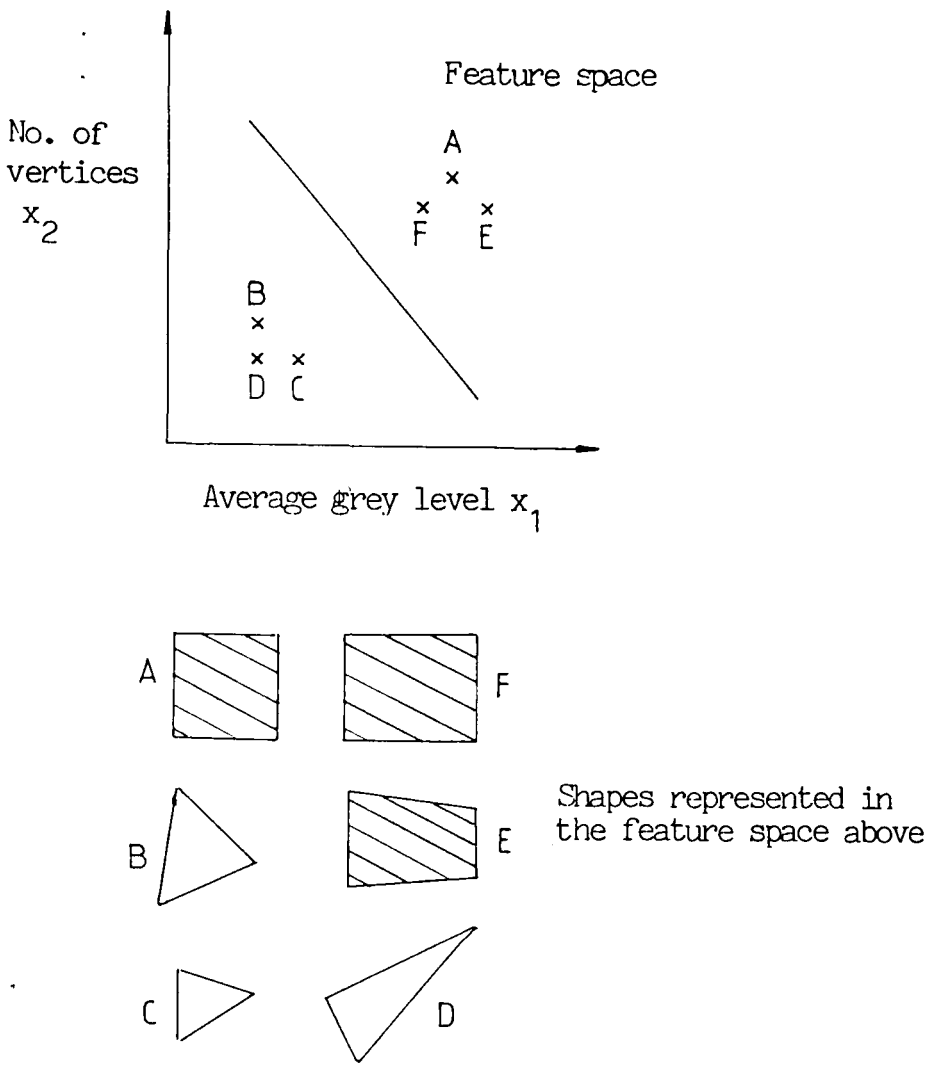


Fig 2.3
Feature space representing the shapes shown.

level. A line can then be drawn between the two `clusters` of points which, if many features are selected and represented, is termed a `decision surface`. In the case above, the line can then be described in terms of an equation of type :-

$$d(x) = W1 \cdot X1 + W2 \cdot X2 + W3 \cdot 1$$

Where Xn are the feature variables, and Wn are the weights, adjusted so that $d(x) > 0$ for one cluster, and $d(x) < 0$ for the other (W3 is included to provide classification of shapes which may lie at the origin if W3 were not included). Thus the pattern can be classed if the feature variables are known. For the case above it is quite simple to choose a set of weights to satisfy the decision criteria. The problems with this process arise when many features are needed to enable a decision to be possible. In this case the selection of weight is complicated and it is not possible to determine whether a set of weights

can satisfy the decision criteria beforehand (Stonham1985). A further problem is found when a linear surface cannot separate the classes of shapes and one has to resort to a non linear decision surface. To enable the application of a linear decision function to the data, a non linear transformation is applied to the feature variables. This further complicates the problem in that the selection of the transformation applied to the data is complex.

A class of machines known as perceptrons (Rosenblat1958) were developed to adaptively discover the weights needed to partition the feature space automatically by the presentation of example patterns within each class. These machines were based upon a simple model of a neuron initially proposed by McCulloch and Pitts (Culloch1943). Unfortunately these machines were shown to be unable to solve certain classes of problems such as parity, connectedness etc by Minsky and Papert (Minsky1969).

For example, it can be shown (Stonham1985) that these devices will not find a solution to the following `EXOR` problem.

$$
\begin{array}{lll}
\text{If} & \left.\begin{array}{l} d(x) > Wi\ 1 + Wii\ 0 + Wiii \\ d(x) > Wi\ 0 + Wii\ 1 + Wiii \end{array}\right\} & \text{class A} \\
& \left.\begin{array}{l} d(x) < Wi\ 1 + Wii\ 1 + Wiii \\ d(x) < Wi\ 0 + Wii\ 0 + Wiii \end{array}\right\} & \text{class B}
\end{array}
$$

No weighting arrangement can satisfy the above equation, where the values 1 and 0 relate to feature values. Only by appling binary logic with specific N tuple mapping to the problem can it be solved (This is covered in chapter 3). A set of patterns relating to the equations are given in Fig 2.4.

In general this method can be slow and unpredictable (the selection of

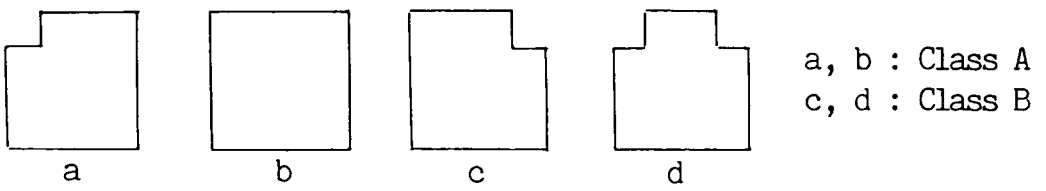


Fig 2.4
Example shapes.

weight and features is difficult), and has the limitations suggested above. However, if a solution can be found, simple hardware can be used to implement it. It must be pointed out that the n tuple processing technique is similar to that used in perceptrons, but is dependent on logical rather than linear functions and is covered in detail in chapter 3.

2.3. Parallel Approaches

This section discusses both `Pattern recognition` and `computational` approaches that can readily be constructed in a parallel architecture machine. That is, the computations required are mainly done on all pixels in the image at once, rather than needing to process each pixel or group of pixels in turn.

2.3.1. Fourier Methods

This recognition approach is interesting because it allows recognition of a shape at any orientation or position in the input image, without the need for brute force searching processes. Basically, the approach extracts the spatial Fourier frequency components of a scene containing the object to be recognised. The amplitude and phase relationships between the Fourier components is recorded. During recognition the object is identified by applying filtering at the recorded Fourier frequencies over the scene. The result is that the object is enhanced within the scene and detection of this `bright` area results in recognition.

The process can be implemented as a fast parallel optical process and is used for military applications where speed and simplicity (leading to reliability) are important. It suffers from the limitation that it only works effectively in simple scenes where only a few objects are present(Pinker1985).

2.3.2. Template Matching

This is one of the most basic methods of pattern recognition. It is based on the use of a `template` or `mask` formed from the example image. In its simplest form the example image is the template. For each shape classification there is a template or set of templates. The comparison between a shape in the scene and a template can be as simple as finding the Hamming distance between the two images (applied to binary images, a measure of the number of bits different between the template and the input image), or a more complex match when grey level images are used. The best match yields the most likely object in the image. In this case a match such as finding the Euclidean distance can be used,

$$\text{distance} = ((X_1 - A_1)^2 + (X_2 - A_2)^2)^{1/2}$$

here X_i and A_i are pixel values in the template and input image respectively. This computation can be slow in large images and simplifications are used with correspondingly less reliability, e.g. the `City block` distance and the `square` distance (Stonham1985).

Problems in template matching relate to the number of computations and the storage requirements needed for all the templates. If we assume that a 256 by 256 pixel image is used, the number of computations

needed to match this against a possible 100 templates of this size is in the order of 10^6 for a binary image, and many more if a grey scale image is used.

Template matching in its simplest form is a linear process. For any given template which is matched on one image, the hamming distance of the template with the image will increase linearly as the amount of noise is added to the image. A non-linear template matching process can be constructed which uses more than one template per class at any one time. Both templates are applied to local areas of the scene. (Both templates are smaller than the total scene size). As the scene degrades due to more noise or distortion, each template will be effected by differing amounts. Thus the sum of Hamming distance of both templates rises non-linearly as the image degrades. It will be seen that this process is very similar the N tuple process described in chapter 3.

The process of template matching is very sensitive to the position of the object in the input scene. For successful recognition the object needs to be in the position it was taught. This either restricts the types of images the method can be used on, or requires many templates to be used - giving views of objects in all positions and increasing the memory used.

Two recent examples of the application of template matching to recognise occluded objects are given in Turney (Turney1985) and in Berman (Berman1985). Both use various methods of optimisation to speed up the process and give it position and rotation invariance capability. Turney uses the Hough transform (See Turney (Turney1985) for reference) as well as a radial template to achieve this, whilst Berman resorts to more heuristic processes, such as looking for invariant

features to centre the object, then using a shift register technique on a radial template to allow rotation independence. Unfortunately both are essentially serial approaches, leading to problems in implementing them in parallel hardware.

2.3.3. N Tuple Methods

N tuple recognisers will be covered in detail in chapter 3. The method is very similar to that used in perceptrons (see above) in that a number of individual processing units look at local areas of the image and combine their results to form one result. They differ in the way the information is processed at each processing unit. In perceptrons data is processed linearly, whereas in the N tuple method binary logic is used at each processing unit. N tupling methods have various attributes that outweigh previous methods. Briefly they are memory efficient, directly implementable in parallel hardware and therefore fast. They are able to solve the recognition problems posed in the discussions on discriminant analysis, but are unable to recognise patterns not in the position they were taught. However, they have been applied to simple scene analysis problems in the past and have been shown to be effective (Dawson1976).

2.4. Main Aspects of the Scene Analysis System

This section draws upon the conclusions of the above discussions to define the various attributes and features that a scene analysis system should ideally contain.

2.4.1. Acquisition of Shapes to be Recognised

Adapting any recognition system to recognise a set of shapes can be a time consuming process if the objects are to be manually encoded into the system. Humans are able to recognise objects by simply presenting the shapes to the viewer. Ideally a recognition system should acquire object descriptions in a similar manner. As well as the advantages of speed and flexibility, in some circumstances the set of objects to be recognised is so large that manually encoding shapes into the system may be impossible. A system which is able to create object descriptions independent of human assistance would be able to perform optimisation between any example patterns and those stored in such a way as to make recognition very reliable.

2.4.2. Parallel Architecture

The use of parallel architecture systems is important in vision where large numbers of computations are needed. Normaly images contain at least 512 x 512 pixels. To perform even simple operations on the whole image would require a very powerful serial processor if the task is to be done in a reasonable amount of time. The use of parallel systems increases speed whilst reducing cost - although at the cost of machine flexibility (Brady1983). Furthermore specific parallel processes should be developed, rather than developing serial processes, then later on considering implementation in a parallel system. The use of this approach is more likely to result in systems that are simpler to implement, more cost effective, and work at a greater speed.

2.4.3. Position Invariant Recognition

A system must be able to recognise an object in any given position in the input scene, at any rotation and size. Whether this is by an active transformation process bonded to a canonical description of a shape in memory, or by a more brute force approach of learning all views of a given shape, is dependent on implementation considerations such as cost, speed and reliability of recognition.

2.4.4. Object Generalisation

In most circumstances objects presented for recognition will not be exactly like any objects previously entered into the system. To allow recognition of shapes outside the set of taught objects, the system must be able to generalise between different pattern sets. This can be characterised by an example of two shapes represented in pattern space. In discriminant recognition a linear surface partitions the two sets of shapes so that objects which lie on either side of the surface can be recognised as belonging to one pattern class. Although no example class may lie in the same position in the feature space during previous teaching as the object being viewed recognition can still be successful. This is an example of generalisation and is a very important aspect of pattern recognition.

2.4.5. Recognition Environment

Ideally, the system should recognise objects independent of lighting conditions and variations in the amount of `noise` present within the image. Noise can either be due to other shapes present within the input scene, or due to random interference due to electrical disturbances, camera optics etc.

2.4.6. Input Image Data

To reduce the cost of implementation, the input image data supplied to a recognition system must be as simple as possible, but not so restricted as to prevent recognition of very similar shapes in a scene. To recognise any object the system could provide a large amount of data about the scene such as colour, grey level data or even stereoscopic views of the image. However, if the object could be recognised using purely binary pixel data the extra information provided to the system would be wasted and involve unnecessary cost. The needs of a system must be accurately analysed in relation to the task before any consideration of the recognition process.

2.5. Summary

This chapter has reviewed the area of scene analysis and outlined the attributes a recognition system should possess if it is to be successful.

CHAPTER 3

N Tuple Pattern Recognition - An Overview

3. Introduction

This chapter describes the N tuple method of pattern recognition, and gives an account of its development.

3.1. N Tuple Fundamentals

The n tuple principal was first described by Bledsoe and Browning (Bledsoe1959) and was originally applied to the recognition of hand and type written characters. This application was widely studied in the 1960's (Bledsoe1959,Bledsoe1961,Bledsoe1962,Highleyman1960) , during the 1970's and 80's studys concentrated on its use in scene analysis (Dawson1976), and face recognition (Stonham1986,Wilkie1983).

The description of the N tuple process that follows will be theoretical, an account of the formal analysis will be given later. The N tuple process described here is that originally given by Bledsoe and Browning (Bledsoe1959) , and remains the predominant method used.

The method can be broken down into 2 distinct processes, the first being `teaching` or evolving operators to respond to a class of patterns, the second being `testing` or recognition of an unknown pattern. Similarly the structural aspects of the process can also be broken into distinct areas, the N tuple sampling, and subsequent logical processing.

Consider the scene in Fig 3.1 containing a single square, N tuple sampling consists of selecting a set or tuple of N small regions from the scene. If the scene had been processed with a television camera and

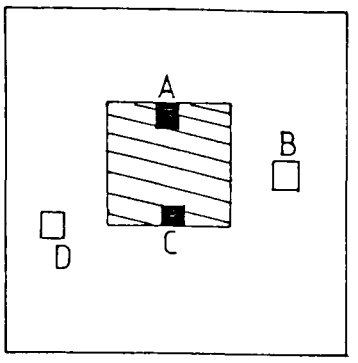


Fig 3.1
Example of one 4-tuple (A - D)
sample from an image.

then converted to pixels, each pixel with a value of 1 or 0, an N tuple sample would then be a group of pixels from the scene. Many such tuples are sampled from the scene and the way in which the samples are taken is important and is termed the `mapping` of the samples. For now a simple random mapping can be considered , where each pixel is randomly selected from the scene, with the requirements that no one pixel is selected for more than one N tuple. Fig. 3.1 illustrates the selection of 4 pixels for one tuple.

The values of each N tuple can be represented in the form of binary logic. If pixel A is white it is written A, if it is dark as ~A. For the sample shown n fig 3.1 the tuple the tuple can be written as ~A.B.~C.D . This indicates that pixels B AND D are white AND, A AND C are dark. The logical relationship between the pixels has therefore been described formally. Other N tuple samples taken from the scene will give different logical expressions. For instance,

~E.F.~G.H

I.~J.K.L

The pattern in figure 3.1 is only one example of a square. If all squares are to be recognised we need to record all the functions they produce. A second set of N tuple samples for another example of a

square may be,

~A.~B.C.D

~E.F.G.~H

I.~J.~K.L

These logic terms can then be grouped with the previous set in the following manner,

~A.B.~C.D + ~A.~B.C.D = tuple 1

~E.F.~G.H + ~E.F.G.~H = tuple 2

I.~J.K.L + I.~J.~K.L = tuple 3

A set of logical functions has now been created which describe the class of objects known as a square. To recognise a given square the image can be broken up into N tuple samples as above and the functions previously produced applied. Each function that is satisfied is counted giving a response for that pattern. On its own this response can be viewed as a measure of how well the image fits the known objects. Normally it is necessary to say whether an image contains one of a set of objects or is one of a class of shapes. To do this a circle, say, can be presented to the system and the logical functions built up as before. These functions would be separate from those produced for the square although they would be generated using the same mapping for the N tuple samples.

A typical set of functions for a circle may be,

A.~B.C.~D + ~A.~B.C.D = tuple 1

~E.F.G.~H + ~E.~F.~G.H = tuple 2

I.J.~K.~L + I.~J.K.L = tuple 3

During testing on an unknown shape, these functions and those for the

square would be applied to the scene and the number of functions within each group (circle or square) which match the input scene counted. The group with the highest number of matching functions indicates the class to which the shape belongs.

For instance, an input image may contain the following samples,

A.~B.C.~D = tuple 1

~E.F.~G.H = tuple 2

I.J.~K.~L = tuple 3

This group contains two terms present in the circle group of functions and one term present in the square group of functions. Thus the input image is more like a circle than a square.

The power of this method of pattern recognition is its ability to generalise, that is, the ability to recognise a pattern as one of a particular class of shapes, even if the system has not been set up to recognise that exact shape. For instance, the set of functions which was constructed from the two examples of a circle can be used to recognise a generalised set of 8 shapes. This can be shown by taking one minterm from each of the three functions for the circle, i.e

A.~B.C.~D = tuple 1

~E.~F.G.H = tuple 2

I.~J.K.~L = tuple 3

These can be used to recognise a circle even though no circle has been presented which contains these functions. In general, the size of the generalisation set, $|G_A|$, is given by

$$|G_A| = K_1 \times K_2 \times \ldots \times K_e$$

where K_j = the number of subpatterns seen by the j`th N tuple sample

during teaching.

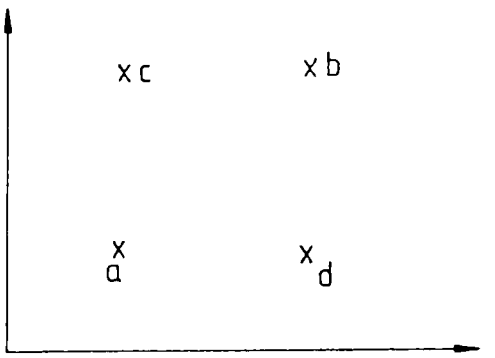
It can be seen that the more N tuple samples that are taken from the scene, the larger the generalisation set. This is looked at in more detail below.

For more details on the generalisation properties of the method see Stonham (Stonham1985).

3.1.1. N Tuple Processing and the EXOR Problem

In chapter 2 it was stated that the N tuple method is able to solve the EXOR problem which could not be solved using a linear classifier. The problem surrounds the need to partition a feature space as shown in Fig 3.2, so that A and B lie in one classification and C and D lie in another. It was shown in chapter 2 that no linear surface can be drawn that satisfies this criterion.

To explain how the N tuple method may be used to solve this problem consider the set of example images shown in Fig 3.3. Each image is made up of two regions which can be either black or white. The labels for each image match those for the ones given in the feature space



x axis = light intensity of left block of shapes in fig 3.3
y axis = light intensity of right block of shapes in fig 3.3

Fig 3.2
Feature space representation of the shapes
shown in Fig 3.3.

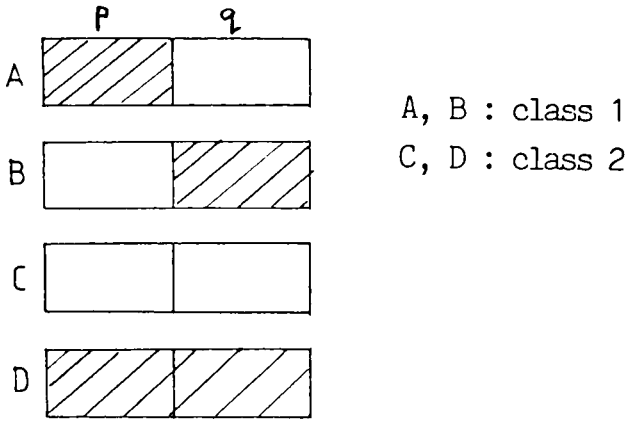


Fig 3.3
Example images for classification.

graph above. Using a specific simple mapping of tuples from a single decoder and two simple discriminators, the problem can be solved. Fig 3.4 shows the arrangement of the N tuples. The discriminators, D0 for class 1 and D1 for class 2, are shown after teaching on the patterns given above. It can be seen that on presentation of pattern A output (ii) from the decoder will be activated. This will recall the associated location in the discriminator. In D1 a 0 is stored and in D0 a 1 is stored. Thus discriminator D0 fires maximaly indicating class 1 is present on the input. Conversely, if pattern C is placed on the input array, location (i) will be accessed in both discriminators. Since location (i) is only at 1 in discriminator D1, the system will indicate the pattern belongs to class 2.

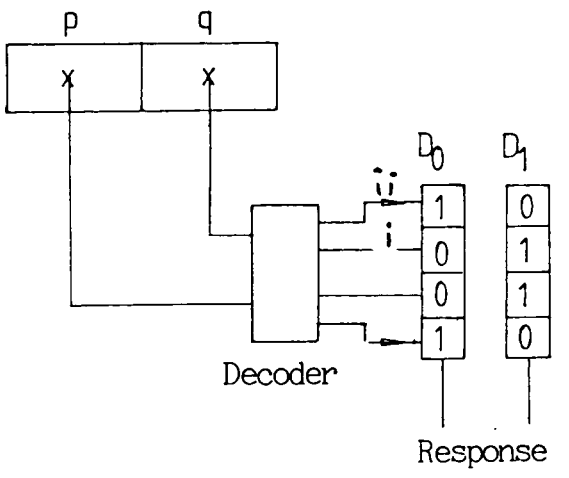


Fig 3.4
N tuple arrangment to solve the EXOR problem.

The system above has effectively learned the following logical functions,

$$D0 = p.q + \tilde{}p.\tilde{}q$$

$$D1 = \tilde{}p.q + p.\tilde{}q$$

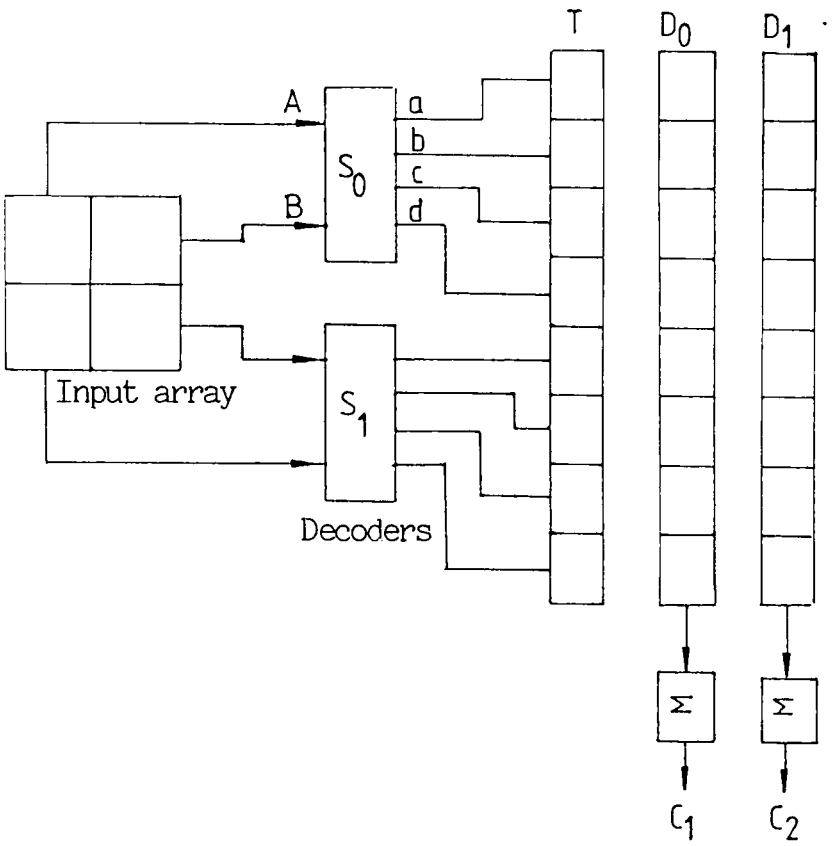
where p and q are labels for the separate areas in the input array shown in fig. 3.3.

The mapping of the tuples within the input scene is of primary importance in this case. If the tuples are not placed so that at least one line from one tuple falls in to region p, and another line from the same tuple into region q, the patterns will not be discriminated since the logical functions shown above will not be possible.

3.2. Implementation of The N Tuple Method

The N tuple pattern recognition method has recently been implemented in dedicated hardware, see (Wilkie1983,Aleksander1984) and Aleksander, Thomas and Bowden (Aleksander1984). The system, WISARD, is a parallel implementation of the N tuple pattern recognition method using random access memories (RAMS) to record and compute the logical functions. To understand how this implementation works consider the diagram in Fig 3.5.

This illustrates the basic elements of the N tuple method as used in WISARD. A 2 x 2 bit input array is shown. The lines A and B form one N tuple sample which feeds a device capable of computing all 2^2 logical functions the tuple can take. This decoder S0 and a second decoder S1 with separate sets of input lines from the scene, temporarily record the functions computed in an array T. The output lines from decoder S0 compute the following functions,



T Tuple vector
$D_0 D_1$ Discriminators
$C_1 C_2$ Discriminator response
$A^1 B^2$ One N tuple

Fig 3.5

Basic N-tuple process

a = A.B

b = A.~B

c = ~A.B

d = ~A.~B

Th two arrays DO and Dl are used to record the state of the array T after each presentation of a pattern in the input array. DO and Dl are termed ´discriminators´ because they are each assigned to one class of shape. There are normally as many discriminators as classes of shape to be recognised.

3.3. Teaching

Teaching the system to recognise a set of shapes proceeds as follows. Here 2 classes of shapes `i` and `j` are considered. An example of `i`

is placed on the input array and the functions produced by the decoders placed in the array T. This array is then logically OR´ed with the array D0 (assigned to class i) and the result placed back in D0. This has effectively recorded the logical functions which relate to the present patterns. Other examples of class i may be placed on the input array and the data set up in T is again OR´ed with the previous data in D0. Patterns in class j are taught in the same way but now selecting D1 to record the decoder states. Teaching is complete when all patterns have been presented.

3.4. Testing

Recognition of unknown patterns is termed `testing´. This process proceeds in the same way as teaching, up to the point of creating the array T. Once this is created it is AND´ed with each discriminator D0 and D1 to discover which logical functions match those previously presented. The result of AND´ing T with D0 is a vector with a number of bits set, where each bit indicates a match with a function set up during teaching. The number of bits set to 1 in this vector is summed and given as a response C1. A similar process is performed on D1 to give a response, C2. To decide which pattern is present on the input, the highest response is chosen.

3.5. Hardware Implementation

In the system described above the decoders and discriminators can be implemented using a commonly available random access memory (RAM). This device contains the necessary decoder, which computes the logic functions of each N tuple sample, and the storage elements needed to record them. In the simple case shown in Fig 3.6, one RAM is used for each N tuple sample. This shows the discriminators and decoders

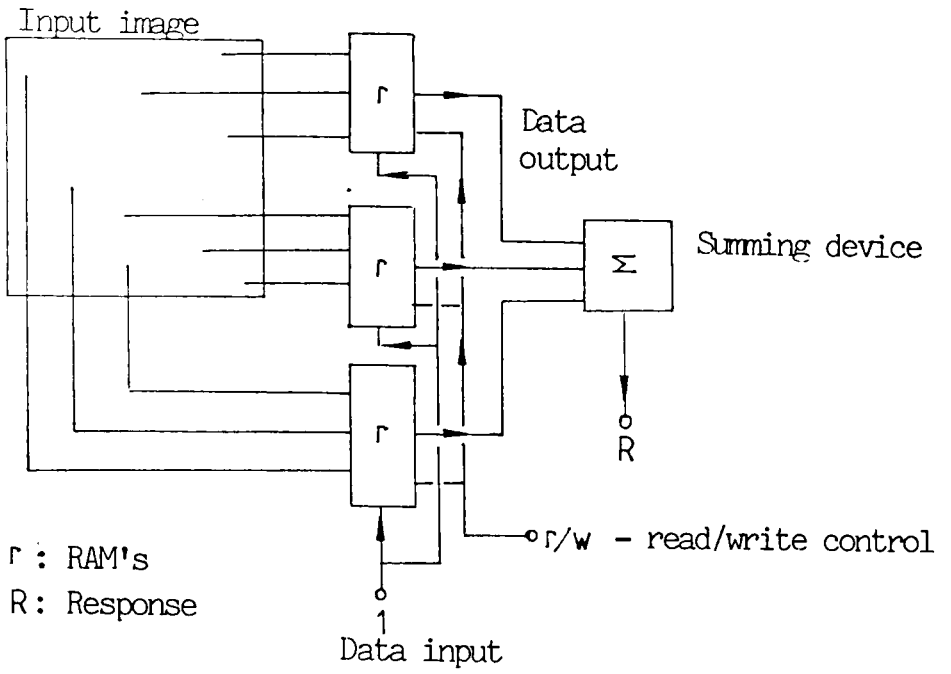


Fig 3.6
N Tuple method as implemented in RAM`s

incorporated into the RAMS. Only one discriminator is shown. The data output lines are summed to produce a response. All RAMs have their data inputs set to 1, so that during teaching a bit is set to 1 at the location addressed by the address lines. A teach or test is selected by activation of the read/write lines on the RAMS.

The ability to implement the N tuple method in the form of RAM´s has enabled the design of systems capable of teaching and testing at 12 operations per second (Aleksander1984).

3.6. Effects of Varying the Size of N

First let us look at the case where N = 1. When this is done the input image is effectively reproduced on the input array in fig. 3.7 During testing, the process of AND´ing the input image with the memory is effectively a template or mask matching process (Stonham1985). The response from a discriminator will indicate how well the input image matches the stored representation (as shown in Fig 3.7).

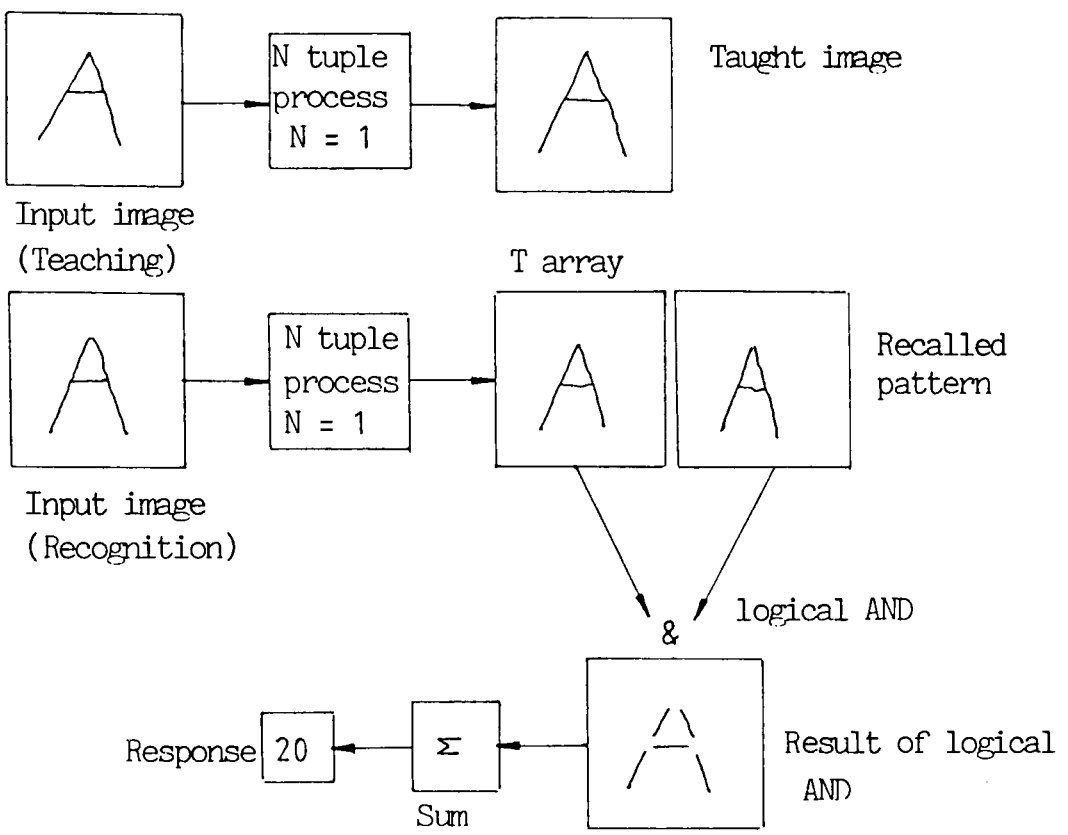


Fig 3.7
Teaching and testing when N = 1

The problem with setting N to 1 can be shown by an example where a Q and an O are to be recognised. If many examples of these shapes are presented, the effective template in the memory will be blurred as shown in fig 3.8 . This will prevent Q or O being discriminated since both patterns would cause the same response on the discriminator outputs. To overcome this problem one possible solution is to use one discriminator for each example of a letter. This process would reduce the method to a nearest neighbour recognition process. Although this is effective as a pattern recognition process, it requires large amounts of memory to save all the templates and lacks the generalisation abilities of the N tuple method.

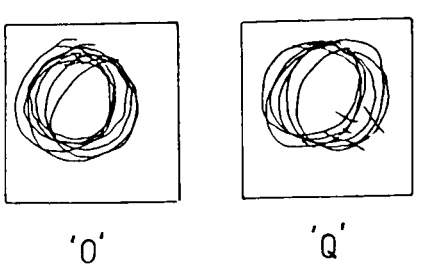


Fig 3.8
Blurred `Q` and `O`

If we now consider the case where N equals the size of the image (S) another problem arises. We now have 2^S functions possible on this single N tuple sample. If the image is of 1000 pixels, the amount of storage needed defeats most practical systems. Furthermore, the effect of setting N = S is that images need to be exactly the same as taught to allow successful recognition. In this form the N tuple method is the same as a library search (Kohonen1977). Here the image is treated as if it were a long binary number, which is used as an address to a location in the memory that contains a flag to whether the pattern has occurred before. Normally hash coding etc. can be used to reduce the storage problem but the lack of generalisation and speed is still a problem.

Therefore, at the extremes of N, recognition is limited normally N is set to a value inbetween these two extremes. The exact effects of various sizes of N have been studied in detail (see Bledsoe1959,Bledsoe1961,Bledsoe1962,Ullman1969). The general effect is shown in Fig 3.9. For low N tuple sizes little teaching is required to obtain maximum possible recognition ability. As the N tuple size increases more teaching is needed to obtain maximum recognition success. Therefore as general rule, the higher the N tuple size, the

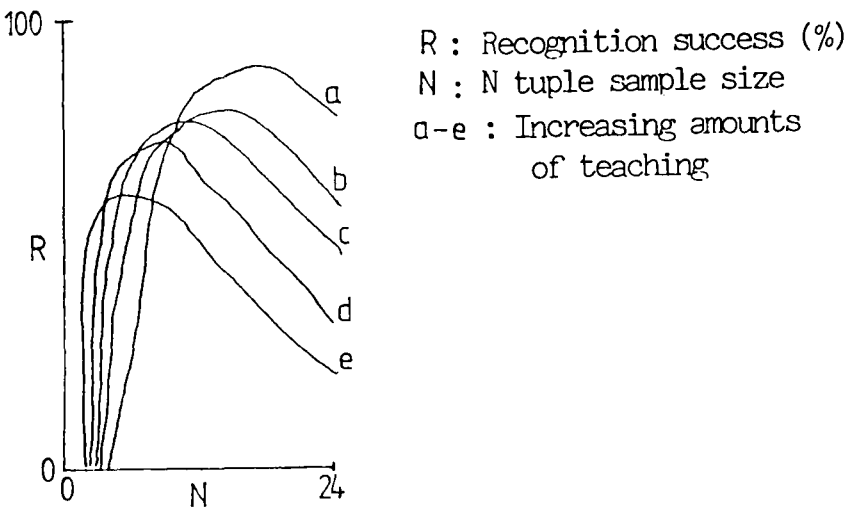


Fig 3.9
Recognition Success for different sizes of N.

higher the recognition success but at the cost of more training.

The concept of saturation is important here. With low N tuple sizes all possible functions on all N tuple samples can be produced by the presentation of a small number of patterns for teaching. It follows from the discussion on generalisation that this reduces the specificity of the system to recognise only the patterns taught. This effect is known as saturation as it causes all locations within the memory of the hardware system to be set to 1.

3.7. Relationship of RAM's with Neurons

The RAM as used in the N tuple method has many features in common with neurons present in the nervous system (Aleksander1983). At a simple level a neuron is an adaptive processor receiving a variety of stimuli on its afferent fibres (dendrites), processing these to produce an efferent response (see Fig 3.10).

In a similar way the RAM used in the N tuple method receives a set of inputs (address lines) and produces a response which is adapted by teaching (See Fig 3.11). However, neurons process linear values and probably function more like an adaptive linear discriminator (Stonham1986).

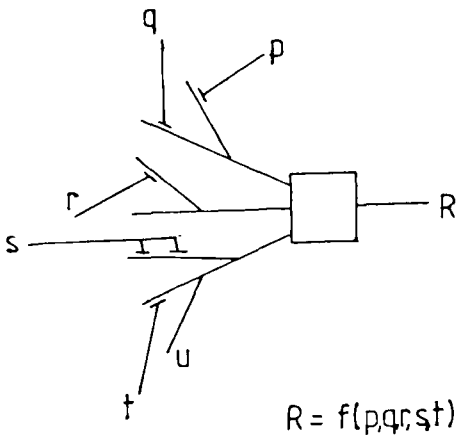


Fig 3.10
Typical Neuron

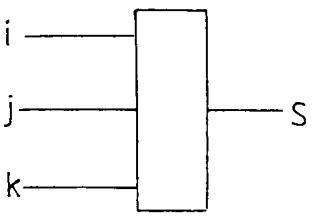


Fig 3.11
Typical RAM.

The N tuple process has other similarities with the human nervous system in that it processes information in a distributed manner, and has the ability to resist local damage. If one RAM in an N tuple processor fails, the ability of the system to recognise patterns will only drop slightly.

3.8. The N Tuple Method as a Feature Recognition Process

The use of N tuples effectively allows recognition of general features present within a pattern. Features in their strict sense are common sub-elements of a pattern, i.e vertices, edges etc. A pattern can be stored by remembering the features that make it up. If each feature was given a label needing less storage space than a feature, memory storage would be saved.

Approaches made by others (Waltz1975) defined exactly what features were to be recognised and stored a priori. The problem with these approaches is that it is not clear which features may be present in a pattern. The process of devising and labelling all possible features that could be present becomes a long and arduous task. The use of N tuples overcomes these problems. Each N tuple decoder effectively codes all allowable states that could appear on its N inputs.

The output of the decoder is effectively a label representing one possible input feature. Furthermore all possible features over the N

inputs to a decoder are represented on the decoder's output as distinct states. The discriminator records which features have been present during teaching. The memory space saved by this process is obvious. If you have a 4 tuple decoder it can represent 16 features and respond to any subset of these. If each tuple was represented in its entirety, it would require 16 x 4 = 64 bits of storage.

3.9. Response Thresholding

The basic N tuple technique can be depicted by the block diagram shown in Fig 3.12. There are various ways in which the responses can be processed to indicate what images are being recognised.

The simplest technique is to indicate which discriminator is responding highest (using an absolute threshold). Since each discriminator belongs to a certain class of shape, this process is sufficient to indicate which object the system is looking at. It is also normal practice to give a confidence value for this decision. This confidence value can be used to determine whether the decision is an acceptable one. If the confidence is above a set value (relative threshold) the decision is passed, if not, it fails. More formally, the process of

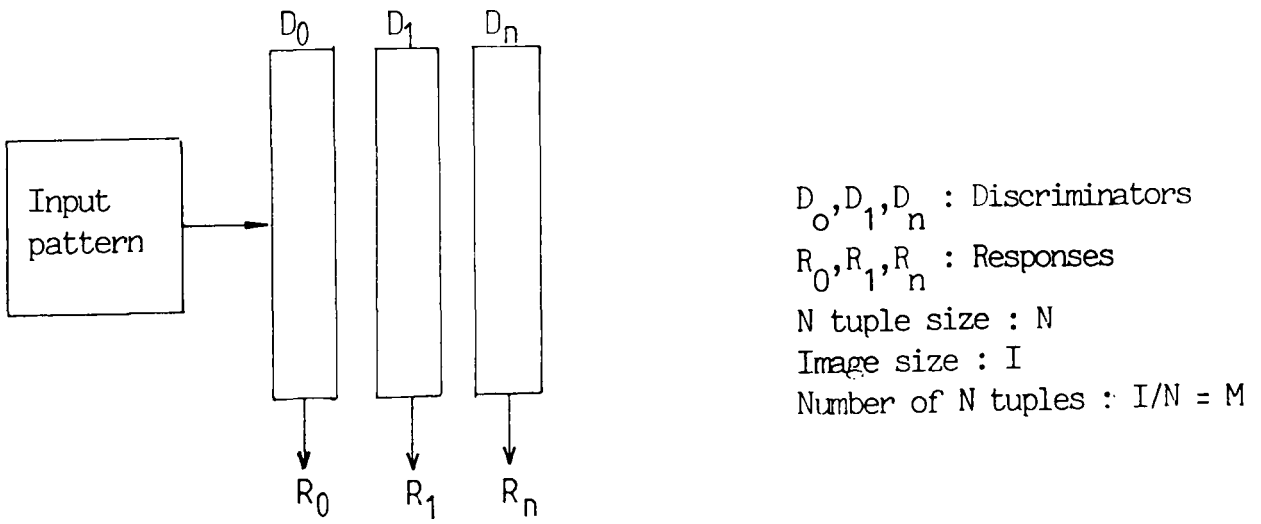


Fig 3.12
N-tuple process.

accepting a classification is as follows.

X is a member of r iff Rr > Ki
and Rr - Rs > Kj

where Ki = Absolute threshold
Kj = Relative threshold
Rx = Response of discriminator x
X = Input pattern
r = classification

Another method of determining the class of the input pattern was described by Bledsoe and Browning (Bledsoe1959). This uses the profile of the responses from all of the discriminators to determine the class of the shape.

If we have a system with, say 4 discriminators taught on an alphabet of `A`, `B`, `C`, and `D`, we can then use this to recognise all of the remaining letters `E` to `Z`. The first stage is to teach the patterns for the letters `A` to `D`, then test the memory on all the other letters noting the response from all the discriminators, such that we now have a data base of responses one for each letter.

On testing an unknown pattern the responses generated are compared with each in the data base the best match wins.

This results in a reduced memory use (only 4 discriminators are needed). Bledsoe and Browning used a variation of this by teaching only basic feature patterns to the memory, one to each discriminator. The result was recognition which was as least as good, if not better when recognising hand written characters.

This approach has been looked at more recently by Stonham (Stonham1986).

This method can be compared to natural systems, where it is unlikely

that each pattern recognised by the nervous system has one cell relating to it. It is more likely that the responses from a collection of cells indicates the label of a pattern. The idea is extended in the associative memory to be described later, in the way the class pattern is used to identify one particular pattern.

3.10. Input Processing Variations

The basic N tuple processing input has four broad areas of variation. These relate to:-

1 Distribution of N tuple samples - Random or regular

2 Distribution of N tuple samples - Local or global

3 Distribution of N tuple samples - Multiple or singular

4 Input data - Binary, grey scale or colour.

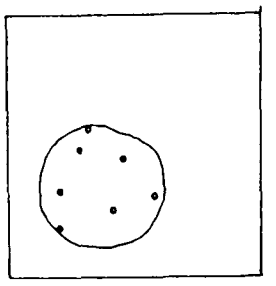
3.10.1. Random/Regular Distribution of N Tuple Samples

In the discussion of the N tuple method, it was indicated that the N tuple samples are randomly distributed over the input array, rather than being distributed in a fixed orderly fashion. The main reason for this is to allow global relational properties present within the image to be recognised. (See next section for fuller description). Bledsoe and Browning (Bledsoe1959) looked at the effects of using different random placing of tuple samples, (Random mapping), and found little effect on the overall recognition of hand written characters. However, it will be shown later that non random mappings may be used in some circumstances to improve recognition success.

3.10.2. Local/Global Distribution of N Tuple Samples

Within the random distribution of N tuple samples it is possible to constrain the limits over which the samples are placed, as shown in Fig 3.13. With local samples, N tuple samples have a fixed small distance between individual sample elements within the input array. With global N tuple samples, the samples are placed at random over a wide area of the input array.

The effects of local and global N tuple sampling can be shown by considering a system which must recognise the circle in Fig 3.14. This shows a cicle present along with a square. Fig 3.15 top shows the N tuple codes generated by local and global N tuple samples when looking at a scene containing only the circle. The image shown in Fig 3.15 bottom gives the N tuple codes generated when the scene contains another shape. Note that the local N tuple sample is not effected by



Each descriminator N tuple sample distributed within a set region

Fig 3.13
Limiting the distribution of N-tuples

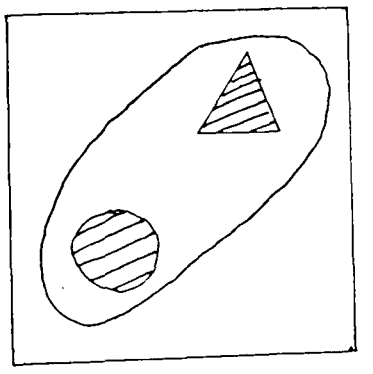


Fig 3.14
Simple scene containing two shapes

the new shape whilst the global sample is. Thus, in general, global N tuple states are corrupted by peripheral shapes leading to reduced recognition success, whilst local N tuple samples are not.

However, the N tuple samples must not be distributed too locally, since this effects the ability to distinguish certain shapes, this is illustrated in Fig 3.16. If class A patterns were taught into one discriminator and class B into another, presentation of class B would yield the responses shown in Fig 3.17 using local N tuple samples. The confidence is very low and, as a result, if the image of class B

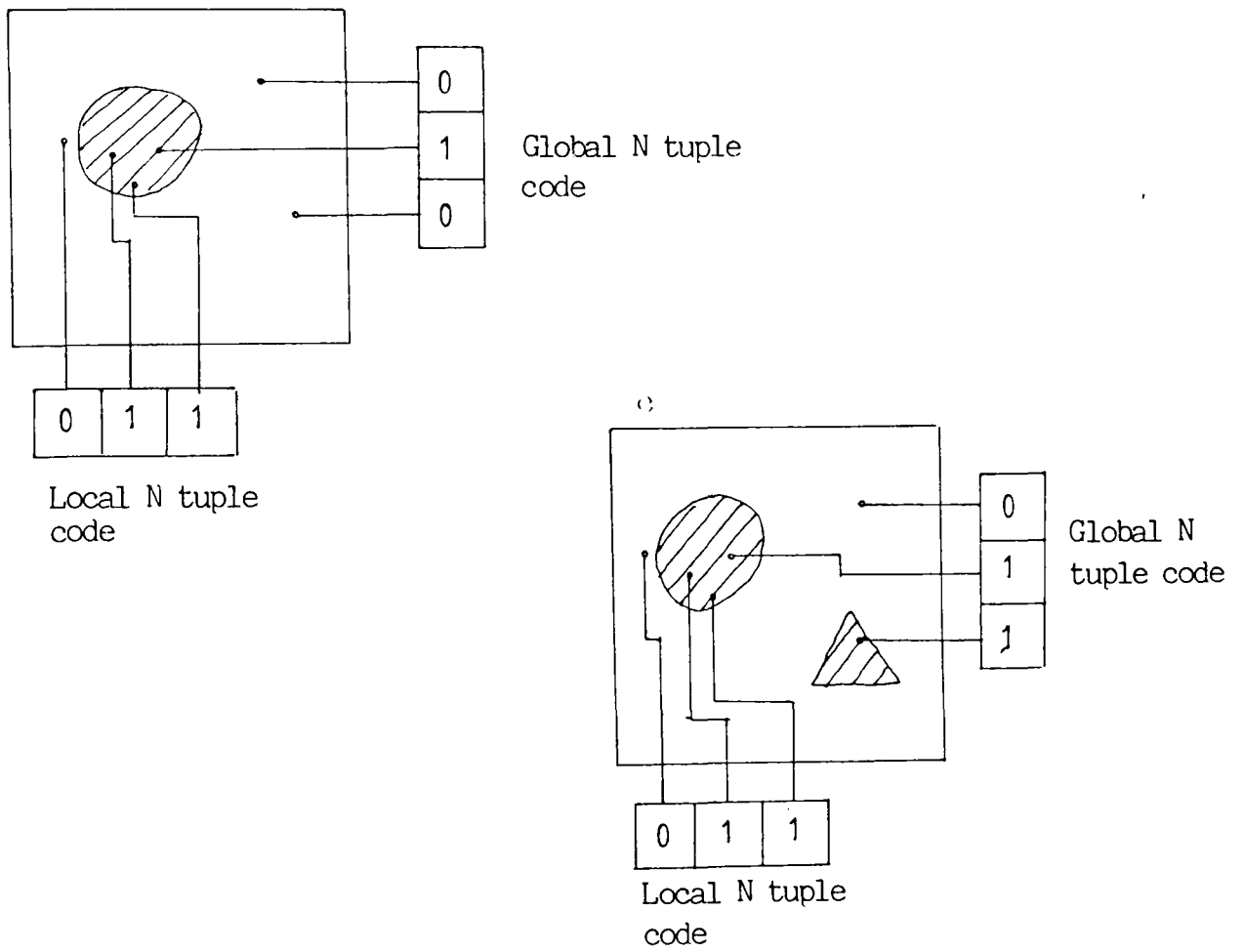


Fig 3.15
Patterns taught into a system containing local and global distributions of N tuples.

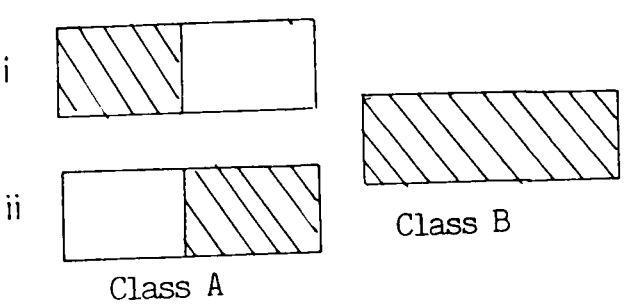


Fig 3.16

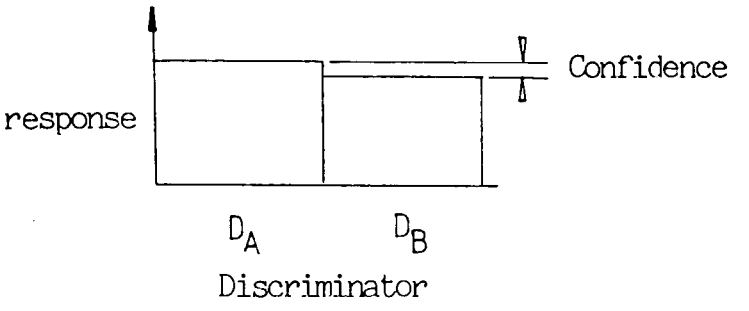


Fig 3.17

was at all distorted, recognition would fail. This happens because all the N tuple samples for the class A discriminator have been taught on an all black pattern - except for those at the boundary as shown in Fig 3.18. The N tuple samples in region A are the only ones discriminating. If the samples are placed more globally, i.e. into the region B, the confidence of recognition between class A and class B would increase.

The outcome of all this is that, ideally, N tuple samples should be placed globally enough to allow high confidences to arise, as in the example, but not too globally to be adversely effected by peripheral noise.

3.10.3. Multiple and Singular Distribution of N Tuple Samples

In normal N tuple sampling each pixel in the input array is connected to only one tuple. This is not absolutely necessary and it is possible to connect many N tuple samples to each pixel. The effect of this was studied by Bledsoe and Browning and found to improve recognition to

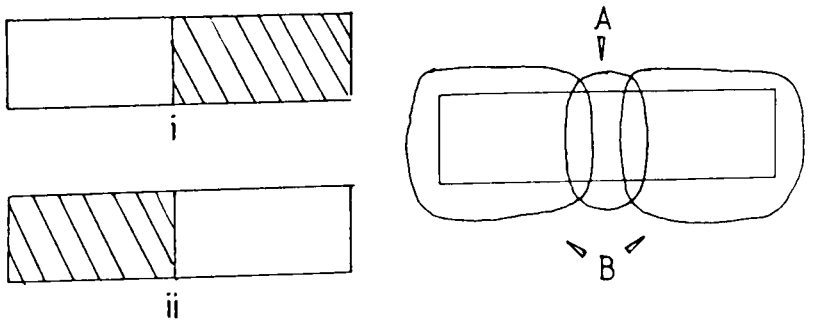


Fig 3.18
Distributions of patterns.

some extent.

3.10.4. Input Data : Binary, Grey Scale, Colour.

Data in the scene can be represented in many ways in the input array. However, the N tuple method requires binary values to be presented. So far only a binary threshold input image has been considered. In some circumstances it may be necessary to use grey scale or colour information, where discriminating information is only available in these modalities. To present grey scale data to an N tuple system some binary representation must be used. Aleksander and Wilson (Aleksander1985) suggest three variations in work done on adaptive edge detectors. These are as follows:-

Binary Coded Grey level

Here the grey level intensity value is put onto the input array as a binary number.

Thermometer Coding

In this case a `bar graph` is used to depict the grey levels - one bar per pixel in the scene.

Bit-sliced Connection

This is similar to the binary coded grey level variation. The first stage is to obtain the grey level values of each pixel in the scene and represent it as a binary number. Then B different input windows are created, where B equals $\log_2$ the maximum grey level in the image. The windows are then filled with the binary pixel values, the first window contains all the lowest order bits of each pixel value, the second window contains the next order bit and so on for all bits in

the pixel values. A separate N tuple system is used for each window. The response during testing is formed by adding together all the responses of each tuple system, Responses from discriminators representing the same class are added to each other.

The conclusions were that bit-slicing was prone to incorrect recognition, but was the most memory efficient method, thermometer coding, was more reliable, and binary coding used a vast amount of memory. For further details see the paper cited above.

Colour input could be done in a variety of ways. Perhaps the simplest is to have an input image and an N tuple process for each primary colour, similar to bit-slicing. For work done in this area see Zissors(Zissors1983).

3.11. Optimisation of N Tuple Sample Distributions

The process of randomly selecting the pixels to which each N tuple sample is assigned has resulted in good overall recognition abilities for the recognition of a general set of undefined patterns. If the pattern set is well defined, in the sense that it is known a priori what patterns are to be recognised, it is possible to manipulate the positioning of tuples within the input field to obtain better recognition i.e. higher confidence levels and less probability of error.

This approach was taken by Aleksander and Stonham (Stonham1974) in recognising mass spectra. In a set of any given patterns it was found that some tuples were receiving the same tuple codes for all classes of patterns. Hence these tuples did not contribute to the recognition of the patterns. These tuples were then reconnected to other areas of the input array. Also it was found that some tuples were receiving all 2^N combinations of patterns within one particular class. These

tuples were not contributing to the recognition of that particular class in any way, and so these tuples were also redistributed. This process resulted in an overall improvement of recognition by increasing the confidence of recognition. The limiting factors in this process are the time taken to instigate optimisation, and the fact that the process can only be carried out when the teach images are rigidly defined.

Optimisation of N tuples was also explored by Bledsoe and Bisson (Bledsoe1962). This was applied to a maximum likelihood N tuple memory (see later) and resulted in improved recognition. Recent work done on optimisation of N tuple samples can be found in Patel (Patel1986).

3.12. Recognition Optimisation by Controlling Pattern Presentation

By careful positioning of the patterns to be taught in the input image, the recognition abilities of the N tuple process can be improved. If the input shape is constrained to a particular position and orientation within the input image then recognition will inevitably be more reliable. Similarly, if the input image is carefully selected in such a way as to present a good sample of the expected variations present within the test set, recognition will improve. No qualitative work has been done on the effect careful selection has on recognition abilities. However, Bledsoe and Browning reported the effects of position and rotation control, and found that it increased recognition success to some extent.

3.13. Position Invariant Recognition

One of the major drawbacks with the N tuple pattern recognition system is its inability to recognise patterns not in the position in which they were taught. To produce position independent recognition a pre processor can be used to align the image before recognition. Alternatively, the object can be taught over a range of positions.

An interesting idea used by Bledsoe and Browning may also overcome the problem of position independence. This is similar to a process described earlier where the response from a set of discriminators, taught on a subset of the shapes to be recognised, was used to identify the unknown shape. If this process is altered slightly position independence may result.

The first thing to do is to select, say, 5 discriminators to be taught. Each discriminator is taught a different shape all over the image, at all rotations. This would result in optimum recognition of the particular shapes anywhere in the input array.

Next select a shape from the set of shapes to be recognised. Place it in the scene and test. The response profile from the discriminators will now be invariant for that object anywhere in the input scene. A record of the response profile can be made for comparison later.

This process is possibly the same as the `Distributed Processing` used by Bledsoe and Browning, but they did not emphasise the position invariance of the process. In addition it is also not clear how the response profile would change in the presence of noise in the input image. The problems of position independent recognition are looked at in detail in chapter 10.

3.13.1. Optimising the Storage of Logic Functions

Most of the above optimisation schemes have concentrated on altering the N tuple sampling and controlling the patterns being taught. The following section considers the effects of alterations to the memory structure used to store the logic functions.

The basics of memory optimisation were described by Bledsoe and Browning. The process effectively records the probability of a particular N tuple state occurring, rather than a logical 1 if it has occurred during teaching, else a logical 0 if it has not. During teaching the number of times each N tuple state has occurred is recorded in the memory. Then this total is divided by the number of teach patterns presented. The result is a value between 0 and 1. This process was reported to give a greatly improved recognition ability when applied to typed or hand written characters (Bledsoe1959). Similar conclusions were drawn by Bledsoe and Bisson (Bledsoe1962).

Various other optimisation processes along similar lines as above were investigated by Bledsoe and Browning (Bledsoe1959). The process which resulted in the lowest error rate was the `Maximum likelihood` technique. The process was similar to that described above, only instead of recording the probability of a tuple being activated during teaching, the logarithm of frequency of occurrence was recorded, giving the tuples not activated a negative value. This process resulted in a far better probability of successful recognition than for any other method tried. However this optimisation should be weighed against the processing needed to undertake such an operation and the amount of memory needed to store all the data. No indication as to the scale of these parameters was given.

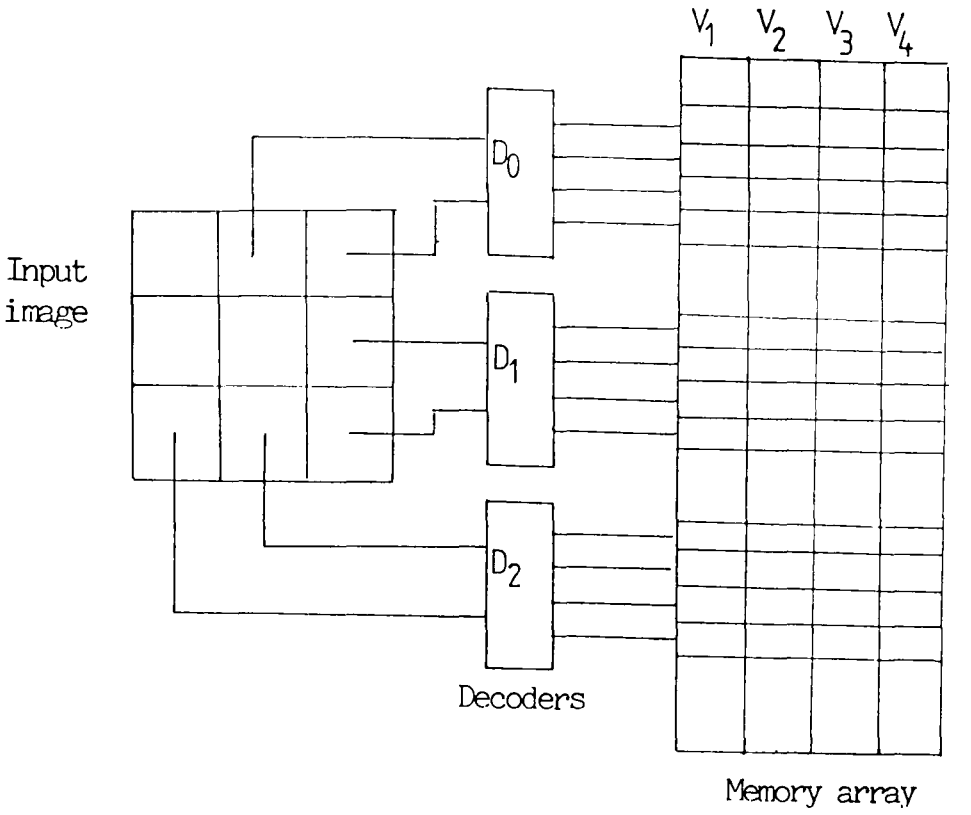
The use of the probabilistic and ´maximum likelihood´ methods was

examined by Ullman (Ullman1969) and found not to be better than the ordinary N tuple method over the complete range of N tuple sizes. Whilst the ´maximum likelihood´ method showed better recognition for low values of N (as in Bledsoe and Brownings experiments), high values of N resulted in poor recognition success when compared to the normal binary N tuple method. These experiments used character data. It can be concluded that the binary storage method is most optimal.

Ullman (Ullman1971) suggested other ways in which memory use could be reduced. Normally P x 2^N bits of storage are needed for each discriminator, where P = the number of N tuple samples taken from the image. Ullman used some interesting data packing techniques to reduce memory whilst retaining performance (by presumably getting rid of redundancy). This process will be described in detail since it is very similar to the methods used in the associative memory to be described later in the thesis.

To appreciate Ullman´s approach a different view of the N tuple process must be considered. The memory system is shown in Fig 3.19. The input image is accessed by a number of decoders with N = 2. These perform the same operation as previously, to indicate the state that appears on the tuple lines. The memory is shown as a bank of binary locations. Each decoder serves 2 power N binary words made up of 4 locations each. Each column of bits is the same as one discriminator (V1 - 4).

The class patterns shown are used in teaching and testing. Consider the normal N tuple process first in which the `normal` class patterns apply. For each class there is one row in the class patterns table. In this case there are four classes C1 to C4.



1	0	0	0	C_1
0	1	0	0	C_2
0	0	1	0	C_3
0	0	0	1	C_4

Normal class patterns
Table A

1	1	0	0	C_1
1	0	1	0	C_2
1	0	0	1	C_3
0	1	0	1	C_4
0	1	1	0	C_5
0	0	1	1	C_6

Packed class patterns
Table B

Fig 3.19

To teach an image a class pattern is selected relating to the image. The image is placed on the input array causing the decoders to indicate the state of the tuples. Each decoder will now have one output line active, indicating one row in the memory array. These rows can now be said to be `addressed`. Now for every row addressed the class pattern selected is logically OR`ed with the data present in memory.

More formally, if any unknown pattern is denoted by X, then the i`th pattern in the r`th class training set will be denoted by Xrj. Furthermore we denote Cr as the class pattern for the r`th class, teaching

can be expressed

Bj(Xri) = Bj(Xri) V Cr where V <=> Logical `OR` (FromUllman1971)

For all N tuples, where Bj represents the selection of the j`th set of $2^N$ words (of length V) within the memory matrix B, and Bj(T) represents the selection of the T`th tuple state within a particular 2 power N set of words. This is the same process as teaching in the normal N tuple system.

Testing is a similar process, but now instead of OR`ing each word addressed memory with the class pattern, a logical AND is performed. If the result of the comparison is the same as the class pattern, then a count is made for that class. A similar comparison is made for all other class patterns.

This can be expressed as,

if : Bj(X) & Cr = Cr then count 1 for class Cr

else if : Bj(X) & Cr <> Cr then no count

for all Cr and j.

The maximum responding class then belongs to the input pattern. This process is also the same as for the original N tuple process. Now Ullman introduces a process known as `Random Superimposed Coding` or Zatocoding after Calvin Mooer (see refs 3 and 4 in Ullman1971).

This allows more classes to be represented for a given number of discriminators than was possible with the original method. In the example above there are effectively 4 discriminators (V1 to V4), thus only 4 classes could be represented and there are, therefore, 4 class patterns in table A (in fig 3.19).

By altering the class patterns to contain Z bits at 1 (Z > 1) rather than 1 bit at logical 1, more classes can be represented. In general for N discriminators we have $\#_N = C^{2N}_{N}$ possible classes. This is done in table B (in Fig 3.19), where Z equals 2, allowing 6 classes.

The process of teaching and testing is exactly the same as before, only now OR`ing and AND`ing all six class patterns. The effect of this is to allow efficient data packing into the memory.

The process is extended by Ullman to another 2 stages. The next stage packs tuples as well as classes in a similar way. These processes will not be described here. The full process has been shown to pack data 4 times as efficiently as normal binary N tupling, while still retaining the same pattern recognition abilities when applied to character recognition.

It is important to note the similarity of this process with that used in associative memory which will be described later on. The use of class patterns is exactly the same and the process of teaching is also comparable. The difference arises during recall since Ullman uses an `AND` process in recall while associative memory performs normal recall as described in the original N tuple process. The response recovered from both processes is then processed to recover the class pattern. There are some cases where Ullman`s process may result in more reliable recognition but no comparative studies have been made.

3.13.2. Overview of Formal Methods of Analysis

It is accepted that an exact prediction of the abilities of an N tuple system to recognise an arbitrary set of patterns is not possible due to the fact that the input data is indeterminate. It is, however, possible to give some indication of the possible responses from an N

tuple system if the pattern variability is known, as Steck (Steck1962) has shown. Roy and Sherman (Roy1967) contrast N tuple pattern recognition with a statistical approximation and also correlate the N tuple technique with a phi processor learning machine (perceptron). Finally, a thorough analysis by Aleksander (Aleksander1985..) has been applied to simple objects, and it was found that the response depended on the intersection of all patterns within the training set and objects presently being tested. Thus a general method of response prediction cannot be derived without resorting to explicit knowledge of patterns being taught.

3.13.3. Summary

This chapter has given a review of N tuple pattern recognition. The basic principal of the method have been discussed together with extensions and enhancements of the process.

CHAPTER 4

Methods of Occlusion Analysis

4. Introduction

The methods of scene analysis described in chapter 2 have two main drawbacks. Firstly they are very computationally expensive (i.e. Guzman1968), and secondly they can only been applied to restricted scenes (i.e Ullmann1983,Boden1977). The occlusion analysis process described here has concentrated on providing a method in which occlusion can be determined in natural scenes containing many unrestricted shapes. The process was also designed so that it could be implemented on a parallel machine, thus providing the speed necessary to process the information present in the large input image (greater than 256 x 256 pixels).

4.1. Input Image Preprocessing

In chapter 1 it was proposed that the system should recognise both `block filled` and `line drawn` images. If we consider a block filled image, a complete `natural` description of the scene is present. However, in `line drawn` images only the outline of the shape is present. A recognition system which utilises the information contained in the body of the shape could only cope with block filled images as this information is not present in line drawn images. Conversely, a system which recognises occlusion using the outline of the shape would fail on block filled images where edge information is not explicit. Although it may be possible to construct a system which takes different approaches to occlusion analysis for each representation, a better solution is to reduce both types of image to a common

representation. This process would incur an overhead in preprocessing the image to one description, but this unification reduces the problem to the recognition of just one type of image, therefore producing a simpler system.

A simple method of edge detection which reduces a grey scale, block filled image to a line drawn image was chosen. The problems of converting a line drawing to a block filled image are much more complex and may not be possible until object recognition has been performed (i.e identification of object boundaries).

Many methods are available to reduce images to edge descriptions using edge operators, see Torre and Poggio(Torre1986). The process is discussed with respect to input processing in general in chapter 9. The description below assumes this process has taken place.

4.2. The Occlusion Analysis Process

Consider the simple scene in Fig 4.1, where one square occludes another of similar size. For simplicity we assume shape (i) is a complete square occluded by (ii), and not a square with a section removed that is occupied by (ii).

One method of finding occlusion is to use motion parallax which

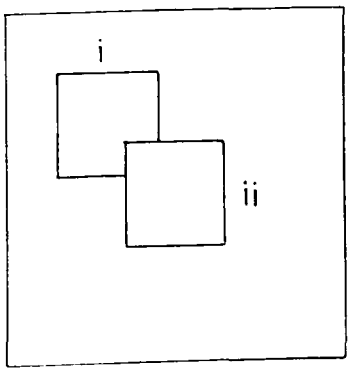


Fig 4.1
Example of occlusion.

detects occlusion by a slight motion of the camera viewing the scene. When this is done the relative positions of objects can be measured and hidden boundaries found. However, this process was rejected as it could not cope with photographs of occluded objects. The use of stereopsis (see Wilson1985) was also rejected for this reason and because of the need to use two cameras, increasing the cost and complexity of the system.

The detection of hidden lines is another method of determining occlusion. If it is possible to identify the parts of an object that are missing, and then deduce that another shape occupies the position of these missing parts, occlusion can be inferred. This process is shown in Fig 4.2. The input image is shown in 1. This is passed to a recognition system which identifies the objects in the scene and recalls

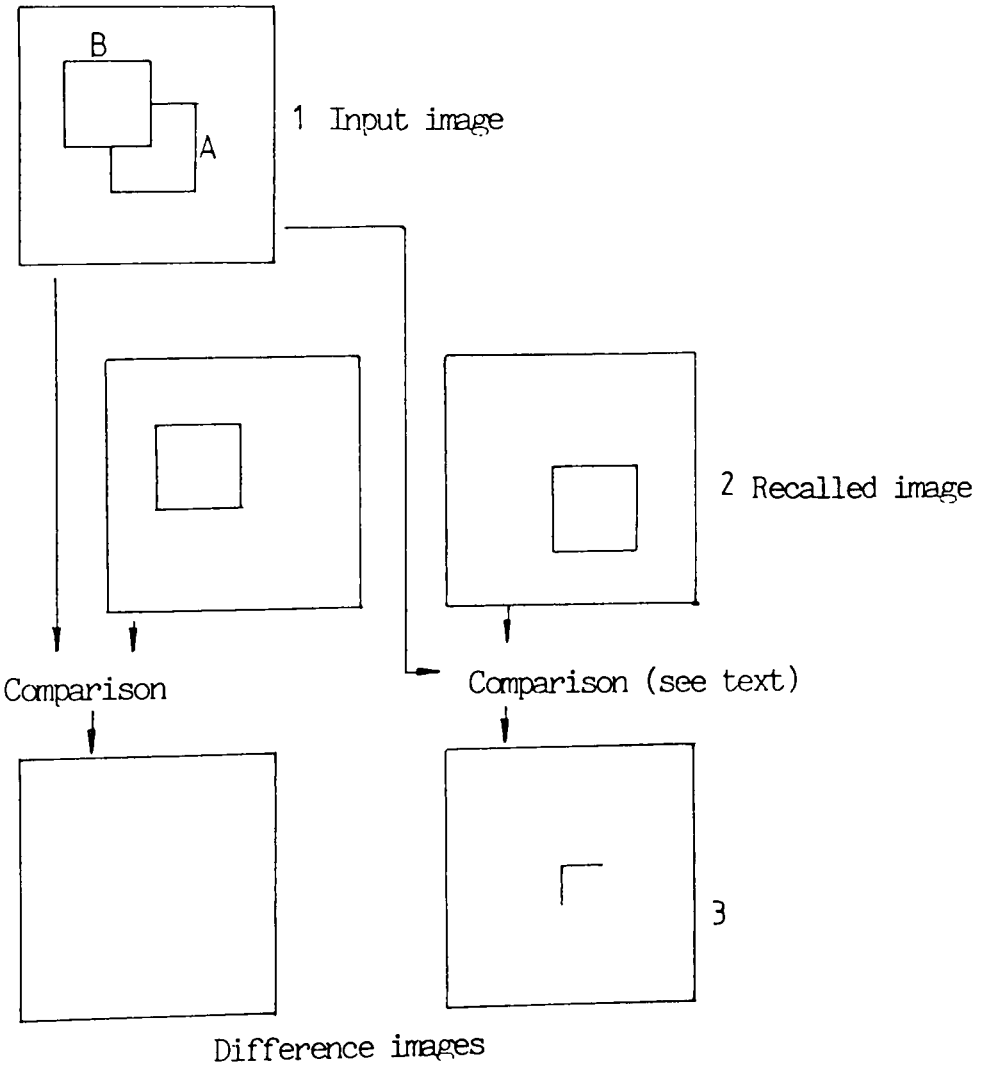


Fig 4.2
Occlution Analysis

their complete shapes from memory, as shown. The recalled images are then compared to the input image giving the position of missing parts from shapes in the input image. The comparison is such that if a point occurs in the recalled image which is not present in the input image a difference is noted, points only in the input image are ignored. By comparing the positions where missing parts are found with other images recalled it may be possible to infer that object B is occluding object A. However, because we are using edge images the part missing from object A will not occupy any part of the recalled image B (compare the difference image in 3 with the recalled image of B above). If this approach were continued with, a possible solution may be to find whether any shape has edges which enclose the part missing from A. Unfortunately this solution would be computationally expensive. Problems would also arise when complete descriptions of shapes were not available, i.e if some shapes were not totally enclosed by edges.

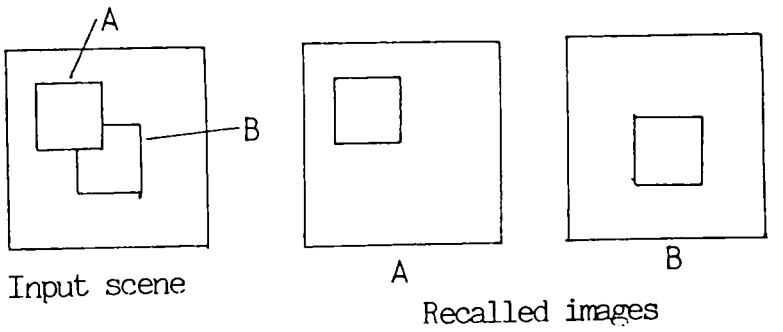
Occlusion can only be resolved at the point where two edges in two different objects meet. At this point a `T` junction is always formed and thus it may be possible to look locally for these junctions. The problem to overcome here is the detection of `T` junctions. One approach is to design a `T` junction recogniser which can first be scanned over the scene to detect `T` junctions present. The system can then go on to infer which object occludes which by using knowledge of how occluded shapes produce `T` junctions. This approach has been taken by many workers (see(Boden1977). Problems with this method lie in the time taken to scan the scene with the `T` recogniser. With large images this can take K^2 operations, where X = Y = K. Also, the design of such operators is complex, although N tuple recognisers, proposed by Wilson (Wilson1985), may be used. Furthermore, after occlusion analysis has taken place the scene would still need to be

processed to find which shapes are present.

The following solution uses a simpler process which combines the recognition of objects in the scene with the search for `T` junctions. We can assume that we have a recognition system which is able to recognise all objects in the scene and recall a complete description of each. These descriptions may be stored in separate image arrays as shown in Fig 4.3..

The process of recognition and recall of the shapes is left to later chapters. Note that the recognition system recalls the shape in registration with the input image so that points in the recalled images will map one to one with points in the input image.

The first stage of the occlusion analysis process identifies possible `T` junctions, by considering only information present in the the recalled images.



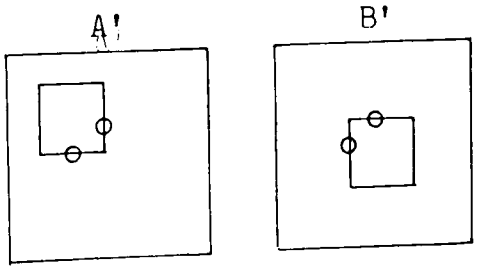


Fig 4.3
Detection of occlusion.

Taking the example shown in Fig 4.3, images A and B can be compared to find at which point vertices in each shape cross. These points indicate occluding junctions. Once identified the system can go in to process these areas locally to discover any occluding information in the following way. Consider Fig 4.4, which shows the image of one junction from the image in fig. 4.3:- The diagram shows a local point of intersection in the edged objects A and B shown in figure 4.3, and the corresponding area in the input image.

The system can infer occlusion from this data by the following process. Same sized regions around a junction are selected from the two recalled images and the input image in fig. 4.4. The points in the image A´ can be compared to those in I´; points at 1 in both A´ and I´ are counted and assigned to a variable pA. A similar process is done for the image B´ and I´ - the value here is assigned to a variable pB. Both pB and pA are then divided by the total points at 1 in A´ (Ta) and the total points at 1 in B´ (Tb) such that pB´ = pB/Tb and pA´ = pA/Ta. Thus the values pB´ and pA´ give an indication as to the number of points in the input image that match with the response image. We can now infer occlusion from these values. If pB´ > pA´, then B occludes A at this point in the image; if pA´>pB´, then object A occludes B.

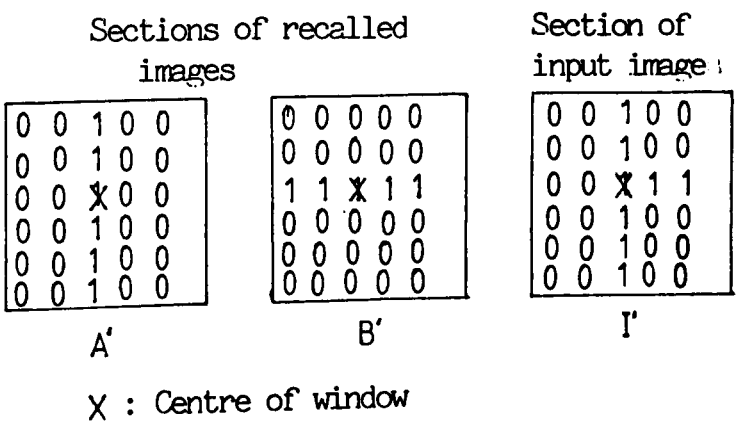


Fig 4.4
One junction from Fig 4.3

This simple process for occlusion analysis has some powerful properties. First of all it is fast in both finding `T` junctions and reasoning which object occludes which since simple comparisons are performed. In addition it can be done on all recognisable scenes and it performs the process locally so that complex occluding shapes can be analysed such as the one given in Fig 4.5.

Unfortunately, however, shapes need to be accurately recognised so that their edges exactly register with the recalled line image. The process can be assisted by further processing to align the edges of the shapes. After a proposed junction has been found the image is scanned within the area where a junction is expected with a `T` junction detector of the type described above. Hence, the possible positions of junctions within the scene can be located.

The input window is then aligned with the junction in the scene that best `fits` the data found locally around the proposed junction in the recognition array (a simple Hamming distance check can be used to find the best fit).

This process eliminates a large part of the misalignment between the objects recalled and the scene data. Once alignment has been done the occlusion analysis process described continues.

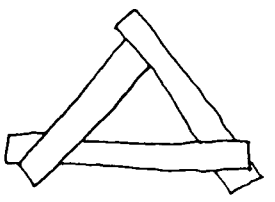


Fig 4.5
Complex occlusion problem.

4.3. Summary

This chapter has introduced the basic occlusion analysis process which uses a novel process to detect `T` junctions. Once detected these junctions can then be processed locally to discover the order of occlusion. The process identifies the need for a recognition system which recalls the complete object from a partial description of the object in the input array. Furthermore, the object recalled should be in the same position in the response array as the object is in the input array. The input image, which may be of `block filled` or `line drawn´ objects, is reduced to a common edge description which simplifies the recognition problem.

CHAPTER 5

A Review of Associative Memories

5. Introduction

The occlusion analysis process described in chapter 4 identified the need for a memory device that could recall a complete description of a shape if given an incomplete description of the same shape. Such a device is an associative memory (see definition that follows). This chapter considers different approaches to the design of these systems, highlighting the strengths and weakness of each in the present application.

5.1. Definition of Associative Memory

The theoretical aspects of associative memory can be stated as follows. It is the need to link patterns such that if a pattern A is paired with a pattern B, pattern B may be recalled at any time by presentation of pattern A. In the case considered here, pattern A may only be an approximation to the original pattern A, in so much as a subset of the original non zero elements of A are set to 0, and original zero elements of A are now set to 1 (considering binary patterns).

In general two types of association can be definedPalm1980, heteroassociation is the process defined above where patterns A and B differ, autoassociation is the case where patterns A and B are the same. (In both cases recall is assumed to be possible on an incomplete input pattern as defined above).

5.2. Different Approaches to Associative Memories

The study of the associative memories is being pursued in two subject areas. Computer scientists and engineers are looking toward a practical realisation of data association, whilst psychologists and neurophysiologists are seeking a model for natural memory processes.

Computer scientists` needs are constrained by practical considerations, whilst neurophysiologists` by what is known of the structure of the brain. As a result approaches differ in many respects. There is also a third section of researchers whos aim is to span both disciplines, utilising knowledge from brain scientists to indicate guidelines to the solution of a practical computer system and visa versa. To some extent this is the position taken by researchers in artificial intelligence and by cognitive scientists.

5.3. Listing and Mapping Memories

The two basic approaches to implementing associative memories have been defined by Palm (Palm1980) as listing and mapping memories. The basic distinctions are as follows.

5.3.1. Listing Memory

In this case each association is stored individually, and sequentially in a conventional computer memory. In the autoassociative case just one copy of each pattern is stored and retrieval can be a trivial process of sequential pattern matching. On each match the Hamming distance between patterns is found. After all patterns have been tested, the pattern with the lowest hamming distance is chosen as the associated pattern. In the heteroassociative case matching and searching is done in the same way but two patterns are stored, the matching pattern

and the key pattern.

The process described above is commonly known as a content addressable memory (Kohonen1984), and has been studied intensely for microchip fabrication (Lee1985), although these systems can be made to recall at high speed through specialised coding techniques (e.g hash coding), they are usually complex and use vast amounts of memory to store the data.

5.3.2. Mapping Memories

The difference between mapping (Palm1980) and listing memories can be described as follows. In a listing memory an association is recalled by comparison to all other associations stored to find the best match. In a mapping memory an association is only recalled by reference to the input key, not by a search; a direct, possibly single cycle, is needed to move from the input key to the associated data.

The strength of mapping memories lies in only one cycle being required for effective recall. This process is identical to that of the associative memory developed in this thesis; an input pattern is presented and directly a class pattern is recalled in one cycle. There is no sequential search.

It will become apparent that mapping memories have the following advantages over listing memories, with perhaps a separate relevance in there correlation to neurophysiologically plausible constructions of human memory.

Speed

Generalisation abilities

Efficient data storage

Fail soft abilities

Little predetermined structure.

5.4. Variations of Mapping Memories

Mapping memories can be divided into two basic groups, continuous, or discrete. Discrete types store data in binary storage elements, whereas continuous types store data in the form of continuous, multivalued data elements. All the mapping memories considered here can be reduced to the basic structure shown in Fig 5.1. Two input arrays and one output array are used. The associated pair of patterns are placed on the key array and the teach array, and the storage elements modified to allow association between the two, such that on presentation of a key array, the associated array can be recalled on the output data lines. The vertical and horizontal wires can be seen to carry data. Where they intersect a modifiable `link` can be made which is implemented as a element. This effectively couples the specific key and teach lines by a certain amount, which may be all or nothing (i.e.

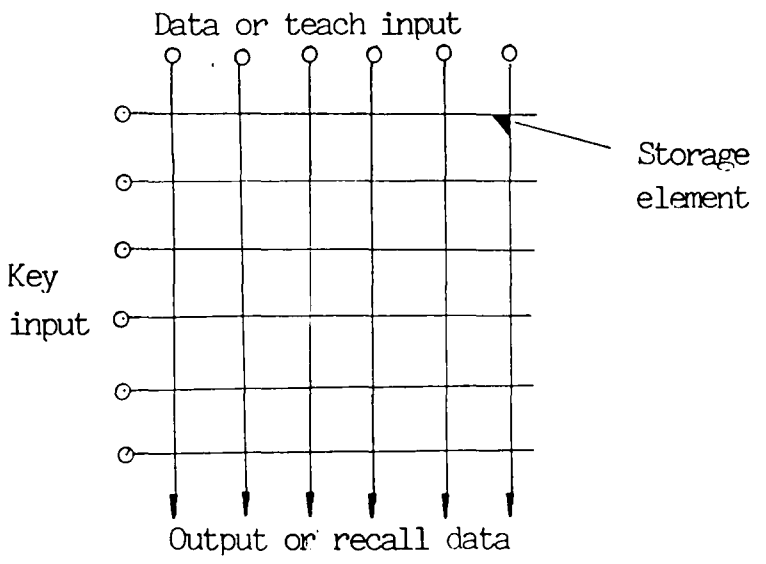


Fig 5.1
Basic associative memory.

binary) or graded (i.e. continuous). On recall the strength of this coupling allows data flowing down a key line to be transferred onto the recall lines by some amount. The sum of all such data on one recall line is given at the output of the memory.

As well as the distinction between continuous and discrete storage elements, there is a further separation between studies on memories with continuous (i.e grey scale) and discrete (i.e Binary) input patterns.

5.4.1. Continuous input and storage

This class of associative memory has been studied in depth by Kohonen (Kohonen 1972,1977,1984,1973). His method can be explained by matrix algebra as follows. The set of links forming the basis of the memory is seen as a matrix M, the key patterns forming the columns of a matrix X and the data patterns as the columns of a matrix Y.

He defines the problem as one in which a matrix M must be found such that,

$$Y = MX$$

The problem is easy to solve if both X and Y are square matrices, since

$$M = YX^{-1} \qquad \text{where } X^{-1} \quad \Rightarrow \text{inverse of } X$$

but it is unlikely that X and Y are square, thus a pseudoinverse (X+) must be found instead, as described by Kohonen (Kohonen1977).

In the case of progressive learning, Kohonen describes a method whereby the original M may be updated for the new pair of associated

patterns, he gives this as

$$M_k = M_{k-1} + (Y_k - M_{k-1} X_k) C_k^T$$

Mk = Mk-1 + (Yk - Mk-1 Xk) Ck^T

The K'th patterns are X_k and Y_k, the term $Y_k - M_{k-1} X_k$ effectively finds the error between what is needed to get Y_k, and what is presently associated. The C_k^T is the transpose of a weighting array, used to move the matrix M nearer to recalling Y_k.

These process are both computationally expensive in time and memory use, although they are an important part of the general work in associative memories.

5.4.2. Discrete Input and Storage

These memories are designed to process binary input images with associated binary storage. They are the type of memory developed in this thesis.

Storage in these memories can be defined as

Image A = Ai

Image B = Bi

Memory matrix = Mij

Mij = 1 if Ai & Bi = 1

else 0

For all i and j

For the case where image vectors A and B are to be associated via the memory matrix M, such that presentation of A will recall B from the memory.

Recall can similarly be defined as

Recall vector R = Rj

Input vector A = Ai

Memory vector M = Mij

$$Rj = \sum_{p=0}^{p=i} Mpj \times Ap$$

For all j

which results in a vector, R, containing a set of continuous (multivariable) elements.

This process is a common way to teach and test the memory. The problem concerning most workers is that of retrieving the original binary teach pattern from the response array R after testing, especially where the pattern being used as the key is incomplete and contains noise.

Many approaches to solving this problem have been made and they will now be discussed.

5.5. Approaches to the thresholding problem

The problem of thresholding the output of a memory is central to all the mapping memories described above. After a test the output will look like that shown in Fig 5.2. The output from the memory has to be thresholded to produce a binary image of the pattern originally

taught. The simple approach of setting a preselected threshold was considered by Gardener-Medwin (Gardener-Medwin1976), He worked towards an understanding of associative memory in man. A preselected threshold could only be calculated for a set of memory parameters, i.e the average number of elements at logical 1 within the teach patterns, and the number of patterns that would be taught. Even with knowledge of the memory parameters it was difficult to set a threshold that would always work (i.e. give an exact recall of the stored patterns).

An important variation of Gardener-Medwins model prevented what otherwise could be an acceptable method of recovering patterns. He used an associative memory in which all links were not possible, that is, not all intersections of the horizontal and vertical wires could form links. This variation is neurophysiologically valid. In computational terms it could also be valid, in that it may be a possible way to reduce storage costs whilst retaining storage abilities.

A recall procedure similar to this was presented by Willshaw, Longuit-Higgins and Buneman (Willshaw1969,Heerden1970) , but here all the links were present. An acceptable threshold level could be calcu-

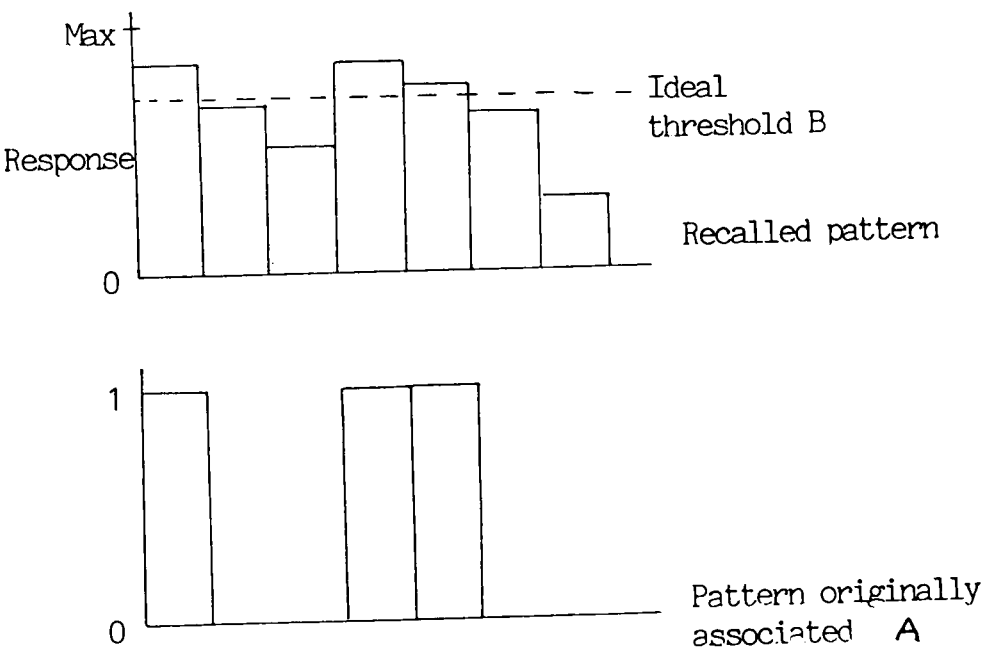


Fig 5.2
Thresholding an output vector in an associative memory.

lated, given a fixed set of network parameters, particularly if the input patterns were restrained such that they contained a set number of elements at logical 1. The output was thresholded at a level x, where x equaled the number of bits in the input key pattern at 1. The result was a memory in which recall could be made if the input pattern was reduced, i.e. some points removed, but not if extra points were added (not original to the input key pattern). This problem is discussed in the design of the associative memory (chap. 6).

Although the approaches described above do not result in error free recall, they have the ability to recall in only one cycle, that is, they only need one test of the memory to obtain a result. This ability to recall data fast is a particular attraction of these memories, coupled with N point thresholding (to be described in chapter 6) they show even greater attraction.

5.6. Iterative Recall

Another approach to recall has been investigated by many researchers. Iterative recall entails many test cycles to `build up` a complete output image. Essentially these are relaxation processes (seeBallard1983). The basic idea is as follows. The image to be used as a key is placed on the input; a test is made resulting in a partial associated image on the output of the memory. This output image is used to create a new key image that is closer to the required output image, but not exactly like it. The new key image is then put through the memory again and another key image is created from the response which is now closer still to the required image. The process continues until perfect recall is obtained.

Lansner and Ekeberg (Lansner1985) investigated this type of process.

They taught an associative memory autoassociatively on many patterns, then used the following procedure to recall,

1) Place key pattern on input (assumed to be complete)

2) Test the memory , obtain response vector r.

3) Select one highest responding element within r that is not a part of the present key pattern, and place a logical 1 in the key pattern at the same position.

4) Re test on the new key pattern.

5) Repeat 3 and 4 until there are N logical 1 points on input key pattern, where N is the expected number of logical 1's in the response pattern.

This process was found to be effective in memories with sparsely taught data. However, if the memories had many patterns taught the recall quality degraded greatly. To overcome this a further process has been introduced.

On each iteration, the response vector r is thresholded at the level X, where X is the average response of all points in r which correspond with points in the key vector at 1. The result is then placed in the key vector.

This has the effect of removing elements erroneously set to 1 on early iterations. This process was found to work effectively. The process of learning heteroassociatively can be achieved by storing a different associated pattern alongside the original, as illustrated in Fig 5.3..

This is an effective in that it can recall patterns reliably, but

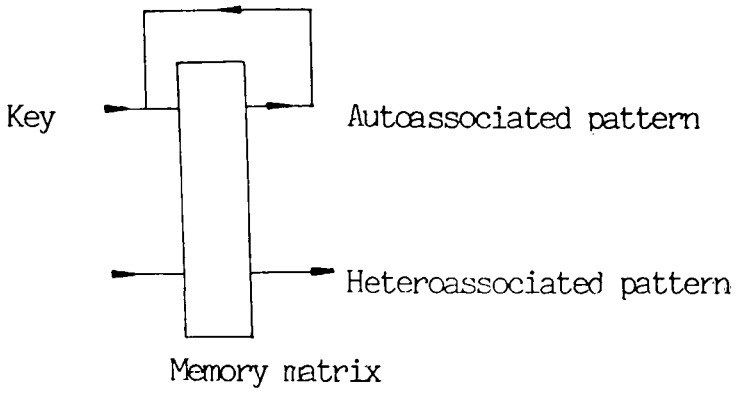


Fig 5.3
Heteroassociation in the memories.

suffers from being slow, needing at least n iterations to recall a pattern, where n is the Hamming distance between the taught pattern and the presented pattern. It also provides little control of memory use, requiring P x P storage elements to construct a memory, where p is the size of the key pattern.

5.7. Implementation of Mapping Memories

Little work has been done on implementing mapping memories in hardware, although Palm and Bonheffer (Palm1984) have developed a multiprocessor system for testing theories about both continuous and discrete mapping memories. Sivilotti (Sivilotti1985) has recently fabricated a Hopfield memory (Hopfield1982) in VLSI, this memory is similar to Willshaws except it uses three state logic and iterative recall.

5.8. Summary

A review of the approaches to associative memory has been given here, with particular reference to binary storage and input systems of the type described in the next chapter. It has identified the central problems of associative memories, in particular the problem of obtaining a perfect recall image of the shape previously taught. The design

of the associative memory described in the next chapter was based upon the mapping memory described above. Although the thresholding of the recalled information needed to be improved this memory can recall data in one operation and could store data distributively - and so has the possibility for efficient storage.

CHAPTER 6

Memory Design

6. Introduction

The occlusion analysis process described in chapter 4 required a memory which could retrieve an image of a possibly incomplete input pattern. The pattern retrieved needed to be in registration with parts of the same pattern on the input array to allow successful occlusion analysis. Furthermore, because many views of the object would need to be stored and an unlimited number of shapes could be presented the memory needed to store data efficiently. The previous chapter reviewed possible memory systems available that could store data associatively. It was shown that a conventional listing memory was inadequate for storing the many patterns required. Although the function of these memories is well understood the problems of storage and speed required the use of an alternative form of memory. The mapping memories have the features required by the system. These memories have a fast search time, and because patterns are not stored individually but distributively the memory used by mapping memories can be reduced compared to listing memories. The exact storage ability of these memories depends on many factors, the most important of which is the process by which the output of the memory is thresholded to obtain correct recovery of the stored pattern. The problems of thresholding are covered below.

As well as providing the speed and storage abilities needed, mapping memories have the advantage of a fail soft ability. That is, if a part of the memory is destroyed, recall of any of the stored patterns only deteriorates slightly. The memories have also been studied in relation to the human nervous system as a mechanism for storing information

(Kohonen1977) , which is of interest to the work described here.

In chapter 2 and 3 the methods of recognising patterns were discussed, it was considered that the N tuple recognition process was a raliable method of recognising the occluded shapes present in the scene. The merits of the process were discussed in chapter 3. This chapter discusses how the N tuple pattern recognition process is linked to a mapping memory and considers the problems of recall and storage control of these memories.

6.1. Using N tuple Logic Functions as an Input to Mapping Memories

A mapping memory can be depicted schematically as shown in Fig 6.1. The patterns to be taught are presented on the teach array and the key array. The key pattern is used in subsequent recall of the `teach` pattern. The memory is taught be actuating the r/w control line to write mode. Recall takes place by presentation of the key pattern and setting the r/w control to `read` mode. The recalled image is then presented on the recall array. The key pattern can be viewed as an indexing pattern as in the listing memories, but here recall is achieved by one `read` from the memory matrix that makes up the map-

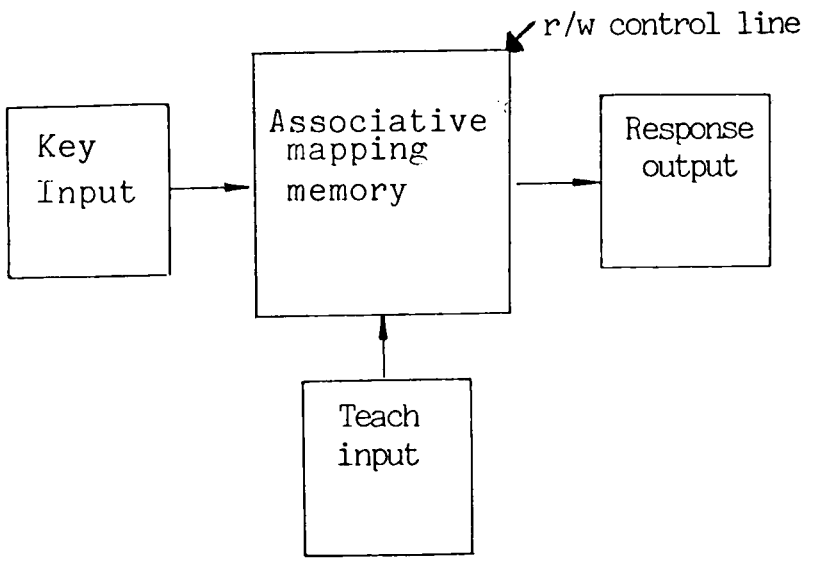


Fig 6.1
Mapping memory.

ping memory.

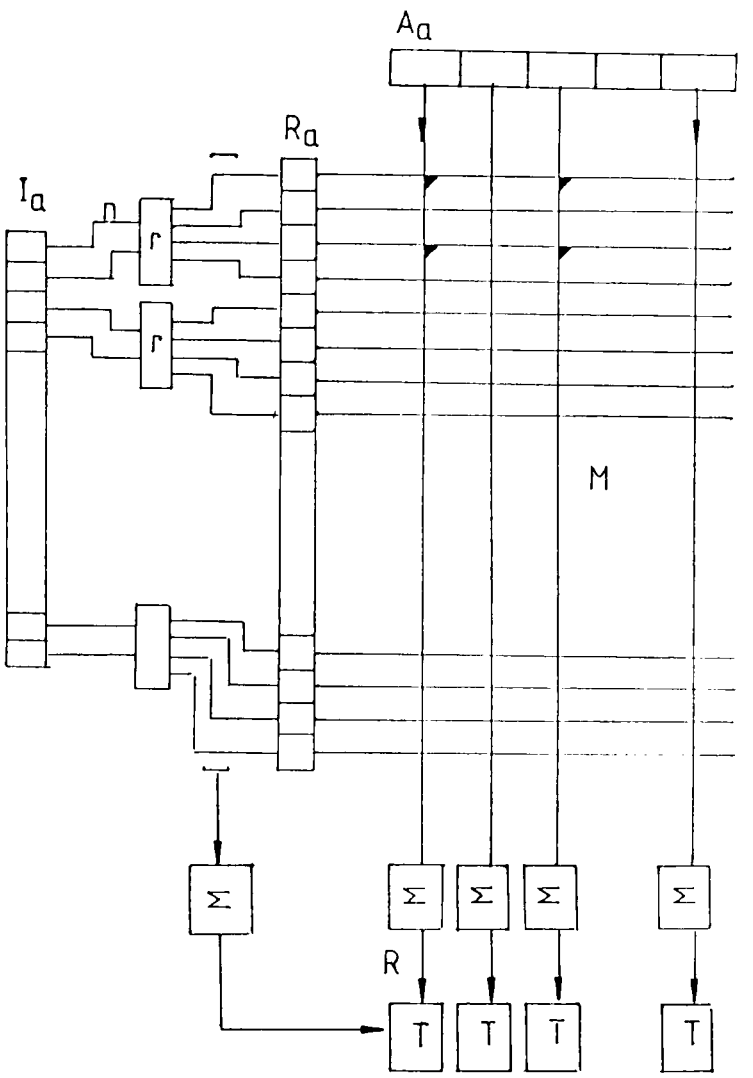
As explained in chapter 3, N tuple processing of an image derives a set of logic functions that describe the input pattern. It was shown how the logic functions can be arranged in the form of an array (see section 3.2 chapter 3) which can record the individual functions present on each presentation of a pattern.

When used with a mapping memory the tuple array which records the logic functions from the scene can be used as the key array shown in fig 6.1. The unprocessed input image can be applied to the teach array during teaching for subsequent recall. Because the tuple array is binary (a 1 representing the presence of one function for one N tuple sample) the binary mapping memory (see chapter 5) is used.

This arrangement, along with the N tuple processing is shown in fig 6.2. The output thresholding process as proposed by Willshaw will be included to illustrate the functioning of this memory (Willshaw1969).

6.2. Teaching the Memory

The process of teaching the memory is as follows, with reference to fig 6.2. The input image is converted to pixels and placed on array I_a, and array A_a. The N tuple sampling of the image takes place on image I_a, via the input lines on n (for clarity the regularly mapped N tuple sampling is shown). The functions on each N tuple sample are then computed by the decoders (r) producing a set of functions over all the decoders in R_a (see chapter 3 for more details of this process). The memory matrix M is the mapping memory shown in fig 6.1. The teaching continues by setting a link in the memory matrix where a row wire and a column wire both at 1 intersect. The row and column wires are driven directly from the arrays A_a and I_a. When all such



I_a : Input array
R_a : Row array
A_a : Associated array
R : Response
T : Thresholding units
r : N tuple decoders
M : Memory matrix
n : N tuple sample lines

Fig 6.2
Details of the mapping memory.

intersections have been set the pattern has been taught and an association been made between A_a and R_a. subsequent patterns are taught into the memory in the same way.

It is useful to add the similarity between this process and the process of storing logic functions in discriminators described in chapter 3. If the first location in the array A_a is set to 1 on every teach the column wire leading from this location is exactly the same as a

discriminator discussed in chap3 sect 3.3.

6.3. Recall from the Memory

Recall of an associated pattern is as follows. The input pattern is placed on R_a as in teaching. The row wires are activated by locations in R_a that are set 1. Now for each column wire the total number of row wires at 1 that intersect with a column wire where a link is set is calculated. The values appear on the output R. Now there is a set of responses that need to be converted to a binary pattern to recover the original pattern placed in the array R_A. To allow this the responses need to be thresholded. The process shown here is that proposed by Willshaw (Willshaw1969). To derive a threshold level the number of elements at 1 in the input array R_a are summed. Values above this in R are set to 1 else they are set to 0. The associated pattern has now been recovered. Willshaws thresholding process works when the input pattern is exactly the same as the one presented during teaching, but fails to recover the correct pattern if the input image contains noise. This will be shown after a consideration of an alternative method of thresholding has been explained.

The next section describes the problems of storage in the above memory and then goes on to discuss thresholding problems in the light of this.

6.4. Memory Use in Associative Memories

The simple associative memory defined above uses a very large amount of storage. This can be shown if we consider a average size example with an input key array and an associated teach array of 256 x256 pixels and set the N tuple size to 4. The size of the memory matrix can

be calculated by multiplying the size of the vector that drives the row wires by the size of the vector that drives the column wires. The column wires are directly driven by the associated teach array which is 256 x 256 pixels in size. The row wires are driven by the N tuple processed key array. The size of this array is

$$\text{(Key array size / N tuple size)} \times 2^N$$

$$= (256 \times 256 / 4) \times 16 = 262000 \text{ bits}$$

This assumes that every pixel in the input key image is connected to one wire in a decoder.

Thus the total size of the memory matrix is

$$262000 \times 256 \times 256 = 1.7 \times 10^{10} \text{ bits}$$

It can be seen that this is a vast memory requirement.

The reason for the large storage requirement is basically because we are associating two very large patterns. We can reduce the storage dramatically if one of these patterns is made smaller. This could be most effectively done by reducing the size of the associated pattern, in essence, replace this pattern with a completely different pattern. This new pattern is called the `class` pattern. So that the original associated teach pattern can still be recalled this pattern is sent to a second associative memory which links the class pattern to the original key pattern. This arrangement is shown in Fig 6.3.

In practice the class pattern is a unique pattern that effectively links the input key pattern with the associated teach pattern by way of the two associative memories. Recall and teaching will now be done on both memories in the same way as one associative memory was taught previously.

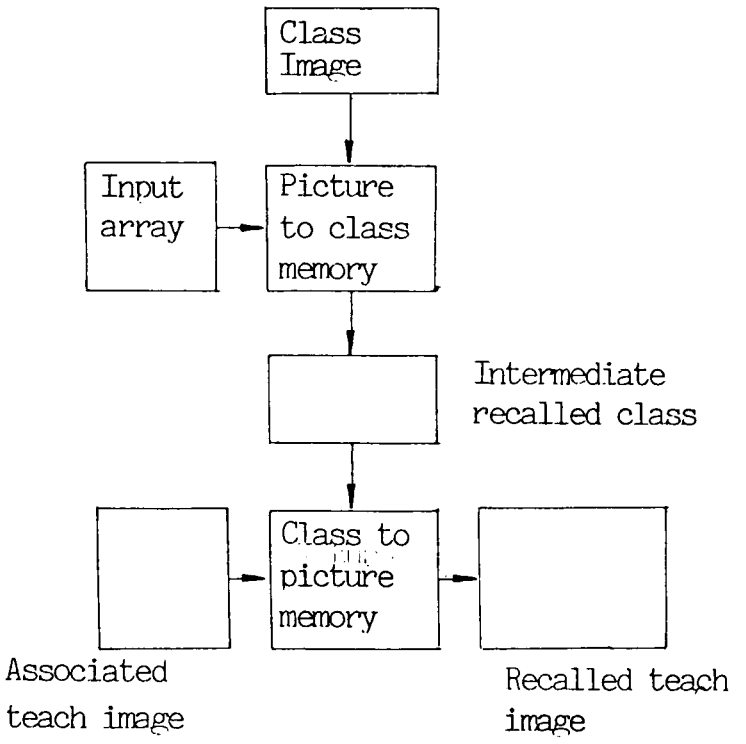


Fig 6.3
Use of a `class` pattern.

To provide the associative properties of the memory the second memory that recalls the teach image from the class needs one of its input patterns tupled. In this case we assume the associated teach image is tupled, the reason for this is covered in the next chapter as it relates to storage abilities of the memory.

By using this two stage system memory system the storage requirement of the memory can be reduced, and furthermore, because the class image is not dependent upon any input image the storage requirement can be accurately controlled.

To illustrate the difference in storage abilities between the one stage and two stage memories consider the following.

The two stage memory has the following storage requirement,

$$k/N \times 2^{N} \times C \times 2$$

Where k is the input pattern size (both patterns to be associated are of the same size), C is the class array size and N is the N tuple size.

The one stage memory the storage requirement is,

$k/N \times 2^N \times k$

By manipulation this shows a storage difference of

K : C x 2 (Single stage : two stage memory)

Thus as long as the class image is less than half the size of the input associated image the storage requirement is less in the two stage memory. In practice the class image is very small and is set to a size dependent on how many patterns will be stored in the memory, this will be discussed in chapter 7.

6.5. Thresholding Strategies

The problem of thresholding the output of the memory to recover a binary image can now be looked at. The two tiers of memory described above have to be considered separately; call the memory which has the input as the key P -> C (picture to class), and the memory that has the class as the key C -> P (class to picture).

On presentation of a pattern to the P->C associative memory, an analogue response forms on the output of memory after a test. This pattern must be thresholded to recover the class pattern, as explained above. Since the class pattern can be any binary pattern, it can be designed so that successful thresholding is possible. The form it should take is to have exactly N bits set to 1, where N is a constant value for all class patterns selected. In recall the response pattern can be scanned for the N highest responding elements. These points are set to logical 1, all others are set to logical zero. This thresholding process is termed N point thresholding. Consider the graph shown in Fig 6.4, which is a graphical representation of a response from the

P -> C memory.

The class array originally taught had 3 randomly placed points set to 1. So, after testing, the 3 highest responding points in the class response image are set to 1 (the reason why this works is considered later).

This process effectively copes with highly changing average response values. A low responding test is coped with as easily as a high responding output.

Now consider the C -> P memory. This system uses the class as its key pattern to recover the image pattern associated with it. It is not possible to use the thresholding process as described above, as the number of points on the response is no longer under control. However, this thresholding process is no longer necessary since we we can now assume the class pattern is exactly the same as the one used as a key during teaching. Instead a simple threshold process is used (the same as is done by Willshaw, see above), where all the points in the response array which show maximum response can be set to 1, and all those below maximum response set to zero.

The recognition/memory device described above fulfils all the require-

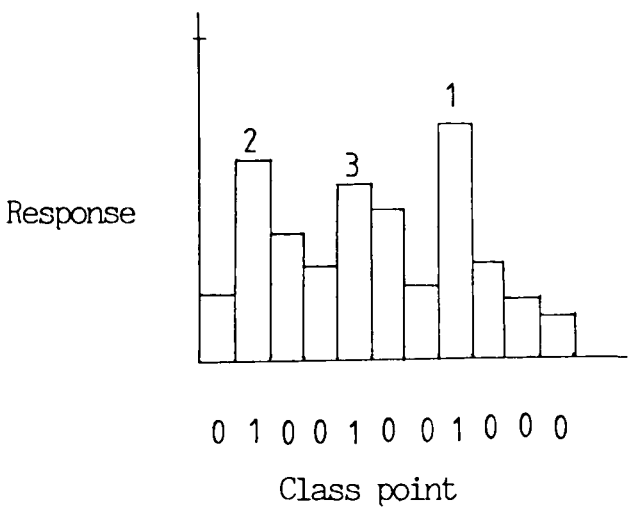


Fig 6.4
Examples of a class response.

ments of our initial specification, i.e

1) Performs association.

2) Uses optimised memory storage.

3) Recall possible on incomplete test pattern.

4) Indicates the class to which the input pattern belongs to.

5) Can be implemented directly in parallel hardware.

The essentials of the design are:-

1) A memory in the form of a matrix of storage elements.

2) Two stage associative store to cut down memory use.

3) A class pattern with a fixed set of points set to 1.

4) Every point in the image addresses a whole class pattern, thus enabling associative recall on incomplete images.

5) Binary storage, allowing construction using RAM´s.

6.6. Factors Relating to the Efficiency of the System

The system so far described has various variables which effect its storage, retrieval and recognition properties. Later on a full review of quantitative aspects of the performance will be given; only qualitative aspects will be given here.

To appreciate how storage is effected by altering the variables, an indication of the statistical factors involved in image recovery must be made and a clear description of operation discussed. Fig 6.5 illustrates the system so far described.

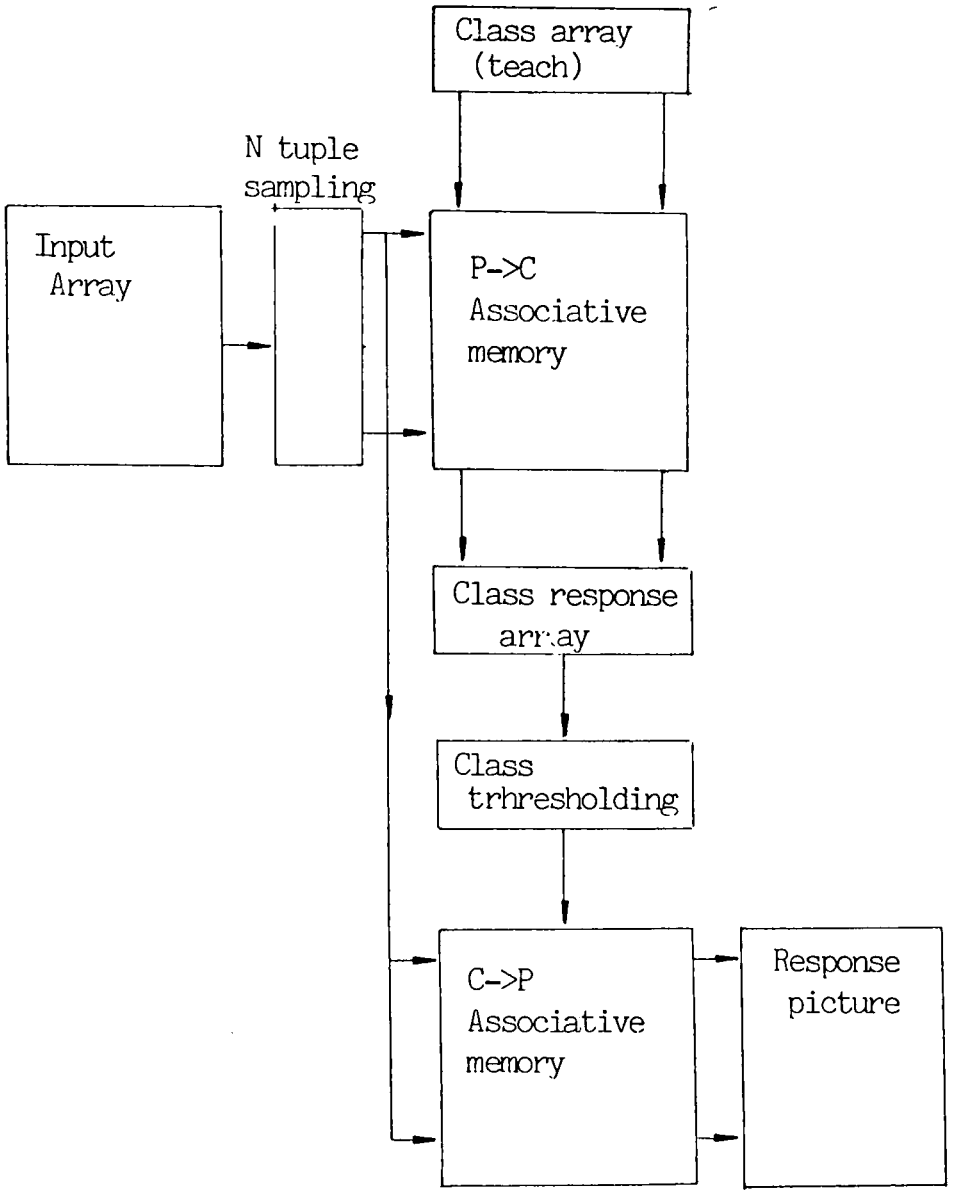


Fig 6.5
Two stage associative memory.

For this part of the analysis, the memory matrix can be considered without N tuple sampling, i.e we assume the input image feeds directly into the memory matrix. The input and class are 1 dimensional binary vectors labeled ´I´ and ´C´ respectively. A simple representation is shown in fig 6.6.

The memory matrix is shown after one teach with ´I´ and ´C´ binary vectors. Coincident 1´s in ´C´ and ´I´ have caused elements to be set to 1 in the matrix. Now consider Fig 6.7 where a second pattern has been taught into the same memory. The class pattern still has two points at 1 but now in a different position. Coincident points in the memory have a 1 written as before.

If we now consider recall as shown in Fig 6.8. Now only image ´I´ is

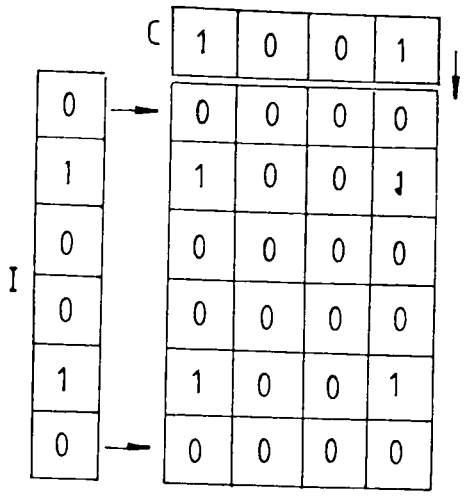


Fig 6.6
Memory matrix after teaching I and C.

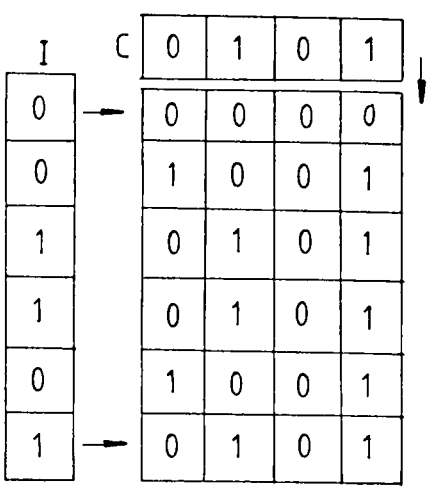


Fig 6.7
Memory matrix after further teaching.

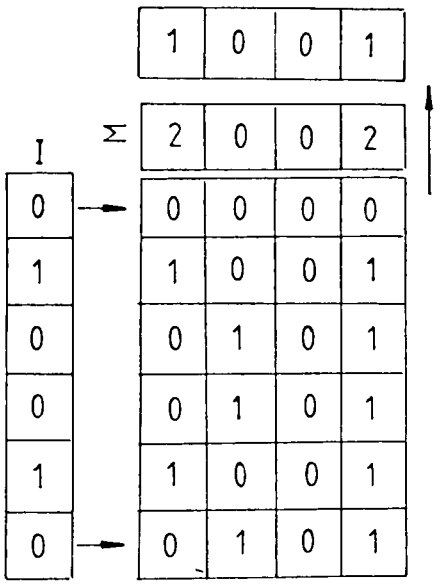


Fig 6.8
Recall of C by presentation of I.

presented, it accesses the memory and for each column any points that are at 1 in the input and at 1 in the in the column are activated. All active points in each column are summed. The class response shown is then N point thresholded. Since 2 points are expected to be at 1 in the class pattern, the first 2 highest responding points are

identified in the response and set to 1. The power of this process is shown by the example shown in Fig 6.9.

One noise point is then added to the input pattern; the effect of this is to make the output response more irregular. N point thresholding recovers the class pattern successfully. It is interesting to note that Willshaws` thresholding processes described earlier would not recover the correct pattern in this case. Willshaws method would produce a threshold level of 3 in the examples shown in fig 6.9 (the sum of bits set to 1 in I). Thresholding the output at this level would result in an incorrect recall in this case. In general, any noise added to the input pattern (such as that produced by an occluding shape) would cause incorrect recall.

6.7. Limits to Memory Capacity

The memory system will fail in two circumstances. Either the image is so distorted as to make recovery impossible, or the memory is ´saturated´, i.e. has reached its limits of storage; this causes a false class response even with a perfect input image.

Saturation occurs when the memory has many of its locations filled.

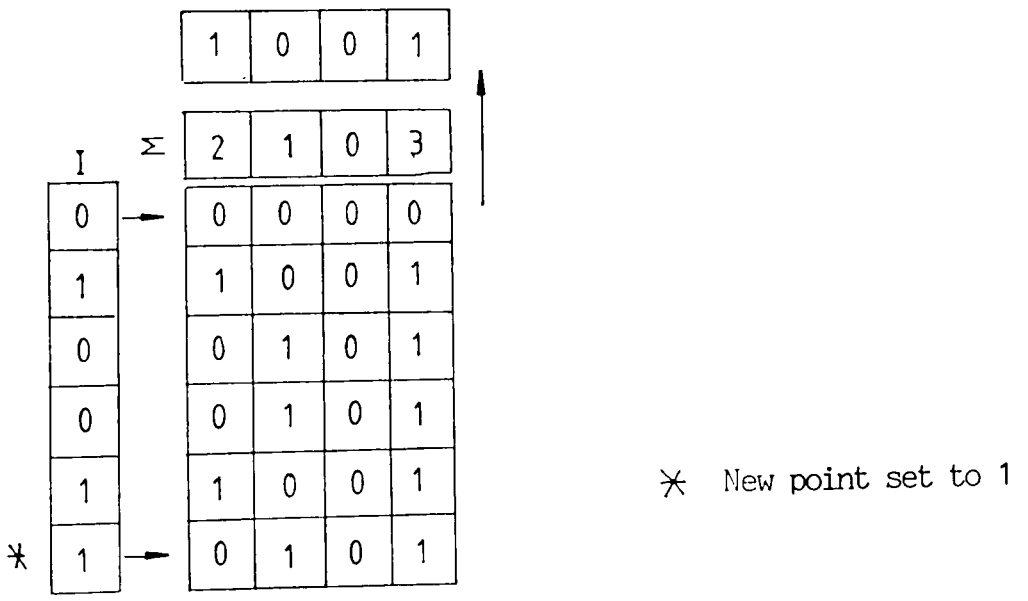


Fig 6.9
Recall of a pattern containing noise.

The process would not result in certain recovery failure unless the memory were 100% full, i.e. every location set to 1. The rate of saturation is dependent on the nature of the input patterns taught. Successful recall is a probabilistic process which is a function of the level of saturation. To illustrate this consider the examples in Fig 6.10. This illustrates a memory taught with one pattern. Recovery results in the correct class as shown. An error in recovery can happen when one of the columns b or c have enough points in them to cause a rise in responses great enough to produce an extra point on the output. A point in the class which is not a part of the original teach pattern is called a ghost point as these do not occur directly from teaching, but are due to internal memory saturation.

Obviously, as the memory is taught more patterns there is a greater likelihood of a ghost point occurring. The actual statistics of this will be examined in chapter 7.

The other factor that effects recovery is distortion of the test pattern. The ability of the system to cope with noise and distortion is a function of the saturation of the system. The more saturated the memory the less distortion that can be tolerated in obtaining an accu-

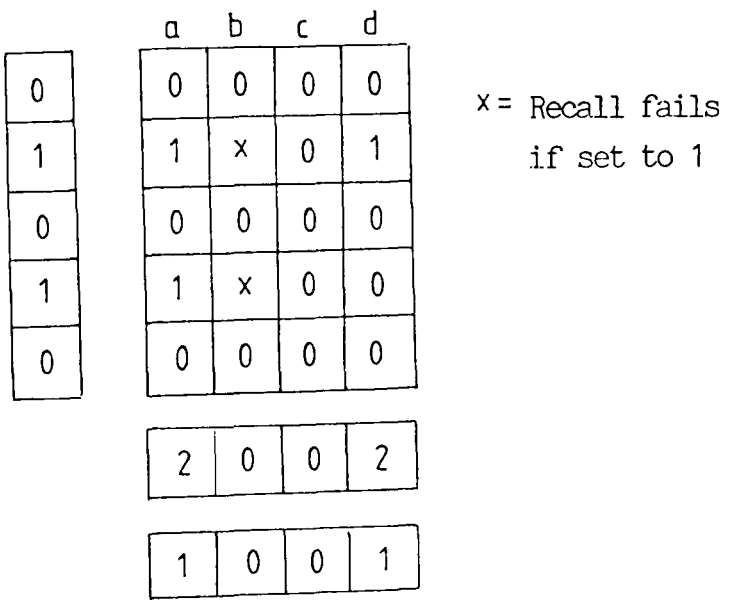


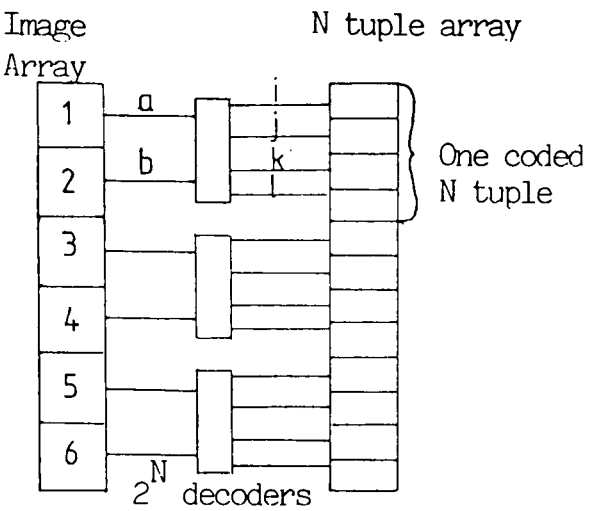
Fig 6.10
Examples of saturation.

rate recall.

Saturation is obviously effected by the size and number of elements set to 1 in the class and input arrays. Basically, if the class array contains 4 points set to 1, and the input array has 20 points set to 1 the memory will have 20 points set to one on each teach. As more patterns are taught into the system, the rate of saturation drops due to the probabilistic nature of the element setting in the memory.

N tuple sampling effects saturation in the following way. It will be noted that if both the input and class arrays have every point set to 1 , one teach will totally saturate the memory. In practice the class array has only a set proportion of elements set to 1 and so saturation is controlled to some degree. In the case of the input array, the process of N tuple sampling also prevents all 1's being presented to the input array and so further prevents saturation. Furthermore, N tuple sampling dictates the total number of points set to one in the memory on every teach, and thus saturation can be controlled exactly. The input pattern can have all points set to 1 and still the memory will only be taught a set number of points. This process of controlling both images presented to the associative memory has overcome one of the main obstacles present in these memories. This is discused in context to other approaches in chapter 8.

Figure 6.11 illustrates how N tuple sampling constrains the number of points set to one in the input to the memory. The saturation rate of the P -> C and the C -> P memories will be equal since the number of bits set in both memories on each teach will be the same. Unfortunately the recall properties of the two memories differ since the P -> C memory input array is large with few points set, whilst the C -> P memory has a small input array with a comparatively high number of



a	b		i	j	k	l
0	0		0	0	0	1
0	1		0	0	1	0
1	0		0	1	0	0
1	1		1	0	0	0

State table

Fig 6.11
Logical N tuple sampling.

point set to 1. This results in the P -> C memory being more reliable in recall than the C -> P memory. This imbalance does not degrade the system since the P -> C memory needs to cope with noise in its input array, whereas the C -> P memory does not. The better recall abilities of the P -> C memory allow it a high resilience to noise, whilst the P -> C memory can be run at a lower average recall ability to achieve the same performance.

Another important aspect of saturation is the randomness of the input patterns taught. If the input pattern remains the same on every teach and a different class pattern is chosen on every teach i.e when a shape presented to the system is the same as one previously taught, this will saturate the memory in local areas, i.e along the rows feeding from each point in the teach pattern at 1. This process is to some extent uncontrollable and is dependent on the type of patterns taught. If the memory matrix is made large enough the process will be less likely to occur. A possible solution is to provide time dependent

removal of elements set to one in rows that are saturated. This is similar to a process of forgetting, and is an area of great interest, although it is outside the terms of reference of this thesis.

6.8. Generalisation Abilities

One of the important attributes of a pattern recognition system is that of generalisation. This process allows recognition of patterns outside the training set. Generalisation can be made to occur in the associative memory by selecting the same class pattern as in a previous teach and teaching a new picture under that class. The problem here is one of teaching strategy and will be covered in greater detail later.

An interesting aspect of generalisation in this context is as follows. If two patterns are taught under two different classes but the patterns belong to the same pattern class and are very similar, will generalisation still occur and how will it manifest itself ?. Consider the patterns shown in Fig 6.12. In the example above the test pattern is a generalisation of the two teach patterns and as a result, the recovered pattern would be a generalisation of the teach patterns.

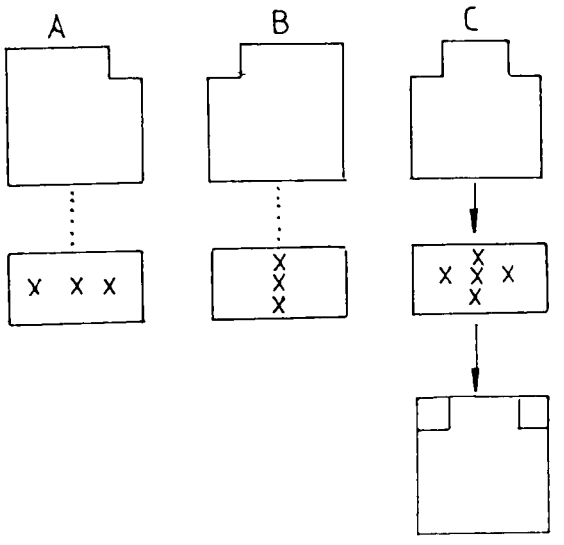


Fig 6.12
Example of recall generalisation.

Note that the class pattern is taught with 3 points set to 1 , but the response pattern has points all responding at the same level. N point thresholding with N = 3 has the effect of setting all 5 equally responding points to 1, effectively recalling both class patterns taught previously. If this is done the C -> P memory will recall both teach images at the same time, i.e if the taught images were A and B, the resultant image would be A OR B, where OR represents a logical or'ing of the points in A and B.

6.9. Linear Input Array

In some circumstances it may be necessary to use linear input arrays to teach and test the memory. For instance, although the occlusion analysis system described is chapter 4 produces a binary image it may be possible to present a more graded description of the scene to the memory in the form of a linear array. This aspect of input processing is covered later in the thesis. The memory described above may be adapted to accept linear input arrays during testing although it is not possible during teaching. Teaching the memory requires binary values to be presented so that the memory can determine whether a location should be set to 1 or 0. However, testing the memory can be done on linear data in the following way. Considering the memory as a normal mathematical matrix and the input vector as a similar one dimensional vector T, recall can be defined as T x M. Thresholding etc. can be performed on the result of this operation in the normal way. Although linear recall will appreciably slow down the system, it may be outwayed by the improvement in recognition performance. This is also considered later.

6.10. Summary

This chapter has described the development of the basic associative memory system as well as its general properties. It is the associative memory that forms the centre of the system to be described.

CHAPTER 7

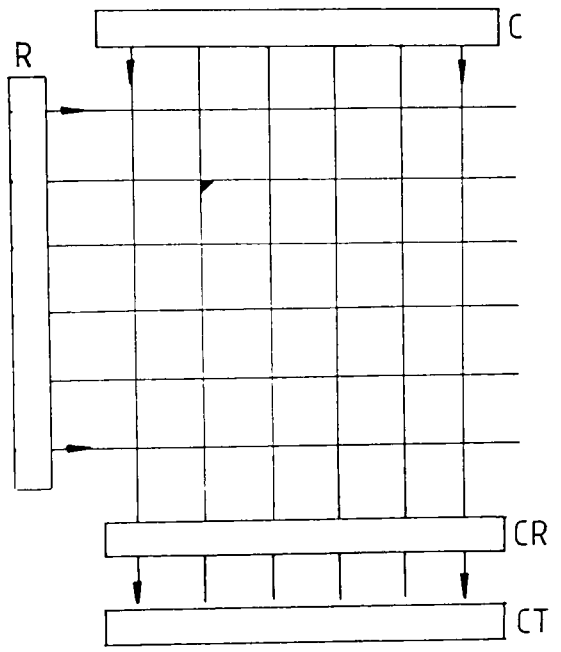
Memory Performance Analysis

7. Introduction

This chapter looks at the performance of the memory system discussed in the preceding chapters.

7.1. Formal Specification of Memory

The associative memory described in chapter 6 has the form shown in Fig 7.1. Consider this in relation to the P -> C (picture to class) associative store. The row vector R contains data relating to the N tuple sampled input image. The column vector C relates directly to the class data. The column response vector contains values generated by recognition and memory access. The threshold column vector CT is the



R : Row vector
C : Column vector
CR : Column response vector
CT : Thresholded column response vector

: Memory link

Fig 7.1
The associative memory.

recovered class after N point thresholding (note, the values of N in `N` tuple sampling and `N` point thresholding are different).

During teaching the N tupled image data is placed on R and the relating class on C. These vectors activate the row and column wires in the memory matrix and where two activated column and row wires coincide a link is made.

Associative recall is performed by placing the N tuple sampled image on the vector R. Then for each column wire, the number coincident row wires where a link is present with a row wire is recorded.

This can be expressed mathematically in the following terms,

If a location in

R vector is expressed as Ri

C vector is expressed as Cj

CR vector is expressed CRj

and one element in the memory matrix as Mij

where $0 < i =< Rmax$

$0 < j =< Cmax$

Rmax = Number of rows

Cmax = Number of columns

also Ri, Cj, CRj, Mij = 0 or 1

Teaching can be expressed as

$Mij = Ri \cdot Cj$ for all i and j

Recall can be expressed as

$$CRj = \sum_{i=1}^{i=Rmax} Ri \cdot Mij \qquad \text{for all } j$$

Followed by N point thresholding on CR, taking the maximum responding N points in CR and setting them to 1.

7.2. Limitations of Recall in the Memory

The memory described above is used back to back with another similar memory to produce autoassociative recall (see chapter 5 and Fig 7.2). This allows an object to be recognised and its recorded shape to be recovered. The memory description above relates to the P -> C memory; the C -> P memory has the same structure but the thresholding of the output is different.

The conditions of failure for the two memories are slightly different. The C -> P memory requires that the exact reproduction of the class pattern occurs during testing. Thus the C -> P memory needs no ability to recall on incomplete data.

The P -> C memory does not have this restriction. The input pattern during testing is expected to be quite different to that on a teach.

Both memories require the output image to be an exact reproduction of

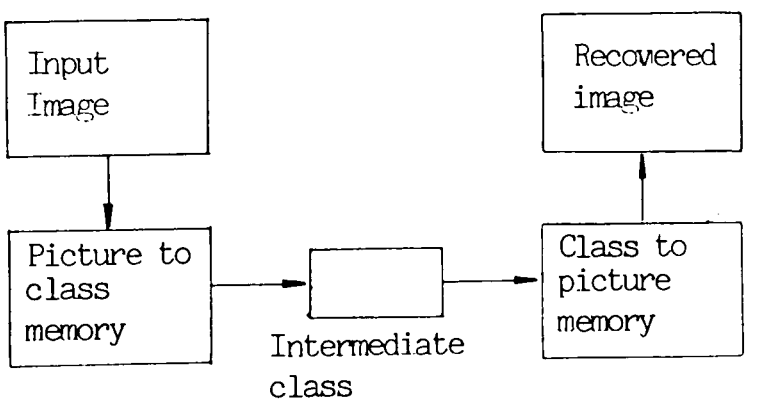


Fig 7.2
`Back to back` connection of the memories.

the image that was taught.

The main failure in both memories is saturation as discussed in chapter 5. To recap, the effect of saturation is twofold. It affects:-

1) The memories ability to recognise incomplete patterns as their complete counterpart is diminished.

2) The probability of extra points in the output array, not a part of the original pattern, increases (`ghost` points).

Case one only effects the P -> C memory as the C -> P memory only matches on complete class images.

As well as predicting the conditions of failure in the memory, the effects on the performance by varying its various parameters will also be examined. A design criteria will be given to enable optimal construction of the memory for specific applications.

7.3. Mathematical Performance Analysis

A mathematical analysis to show the effects of various parametres on memory performance will be given first, followed by empirical investigations to verify the analysis. As described in the previous section, it is intended to describe errors of the output from the memory due to saturation. These errors are probabilistic, and they occur in the following way. Consider a pattern being placed on the input array R as shown in Fig 7.3. This activates the row lines in the memory matrix M. Considering one column in the matrix, the active row wires will coincide with the column where a location may or may not be set. If this column wire was one set in a previously taught pattern, all row wires active during the test would coincide with a link that is set. The number of active rows coinciding with locations that are set

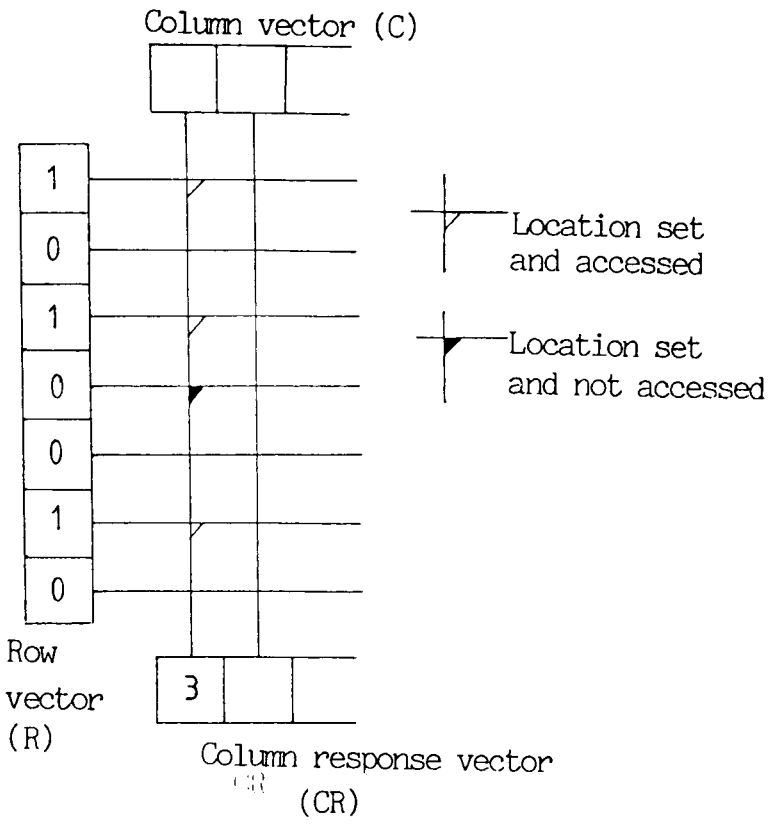


Fig 7.3
Memory access.

to 1 is recorded in the response array at the position representing this column.

The response on the output is then thresholded using the N point thresholding on the class response output, and thresholding at p in the picture response array, where p in this case is 3.

Now consider the case where the column in the matrix was not activated during the teaching of this pattern. The possibility that the active rows will coincide with a set location is purely random. The probability that enough coincidents will occur to allow the total to go above threshold will depend on the saturation level of the memory. Obviously with many locations set it is likely that enough coincidents will occur.

For analysis, the case where a perfect input image is presented to the memory in any test i.e. the complete pattern that was taught shall be considered. This restriction simplifies the analysis. Cases where the input pattern is not exactly the same as the one taught lead to somewhat intractable probability equations, see the discussion on this

below.

The basic form of the analysis is as follows. The use of a perfect input pattern as described above means that the response pattern will be perfect when the memory is well below saturation. Above saturation errors will manifest themselves as `ghost` points, not belonging to the original pattern but created by `internal` effects on the memory.

The process causing ghost point production is explained in Fig 7.4.

The probability that a ghost point occurs is dependent on the saturation level of the memory, since this determines the probability that any one location selected in memory at random will be on. Once this probability P has been found the probability that the N active rows in the test image will coincide with a complete set of these locations is P^N. From this the probability of any column responding with a ghost point can be found and consequently the probability of failure after a set of images has been stored. A fuller description accompanies the

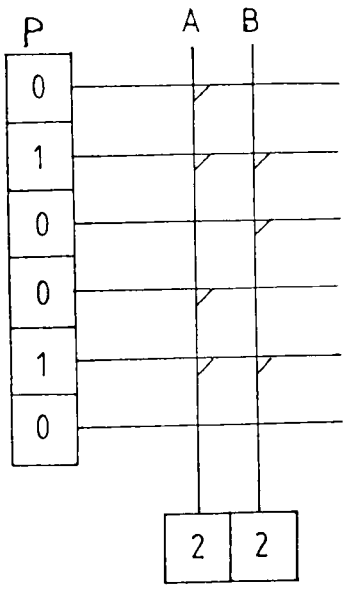


Fig 7.4
Ghost point production

Pattern P was taught when B was active and not A, thus the pattern should only activate B maximaly. Due to coincident points in A, this column also responds maximaly causing a ghost point to occur with a resultant failure in recall.

following mathematical analysis.

7.4. Conventions

The following are used throughout the analysis :

	P->C	C->P	Label
Column vector	Class	Picture	CV
Row vector	Picture	Class	RV
No. elements in CV	Size of array		H
No. points in CV	Active elements*		N
No. elements in RV	Size of array		R
No. points in RV	Active elements*		NI
Probability of failure			P
Number of images stored			T

* : For the class array, the number of active elements is the N points selected within the class to represent the class pattern. For the picture array, the active points depend on the N tuple size (see section on input processing chapter 9 to see how this relates to real input image size).

7.5. Saturation Level of Memory After T Teaches

The analysis starts by predicting the number of points set in the memory after T teach iterations. The full analysis and approximations are given in appendix 1. The equation derived is

$$s_t = H.R\ (1 - (1 - (N.NI/H.R))^T)$$

Where s_t = number of the points set in a memory of H x R points after T teaches.

The probability of selecting an active location in the memory is given by,

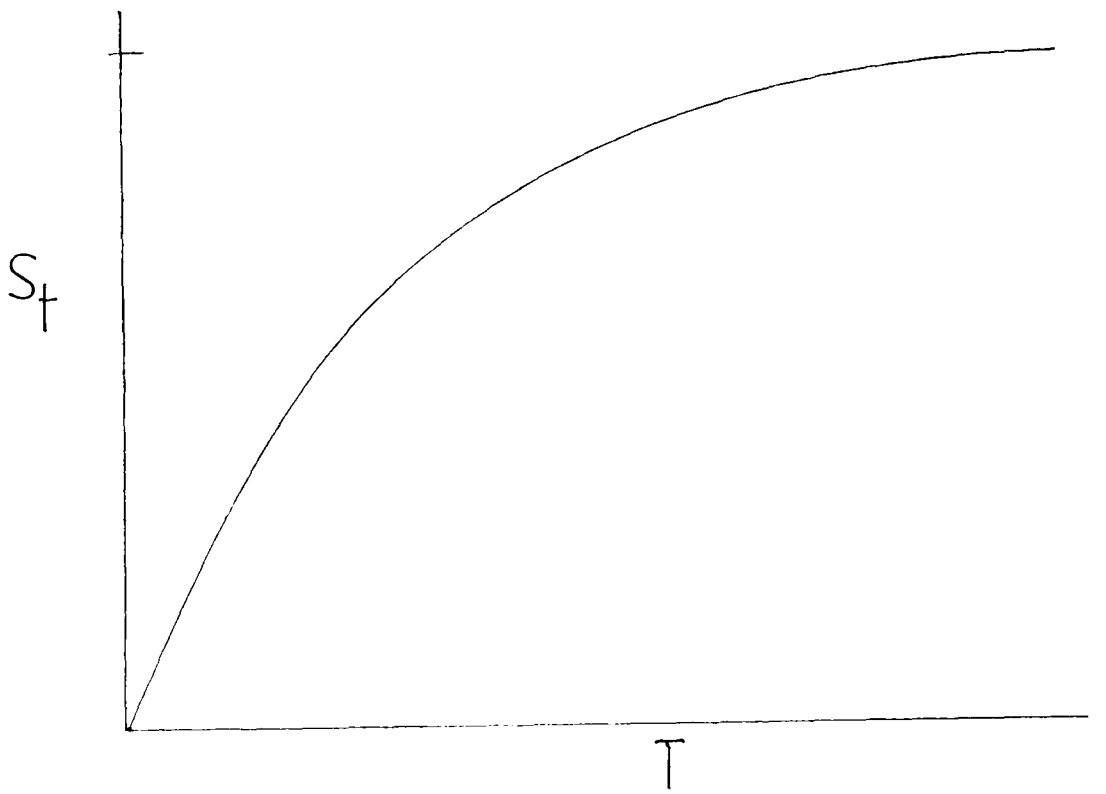
$$\frac{S_t}{\text{total memory locations}} = \frac{S_t}{H.R}$$

$$\Rightarrow p(\text{active location}) =$$

$$1 - (1 - (N.NI/H.R))^T \qquad (1)$$

7.6. Probability of a Corrupted Output Pattern

Its is now possible, using the above equations, to predict when a recovered pattern will not be an exact reproduction of the original. The probability will reduce as more patterns are stored.



Graph 7.1
$s_t = H.R\ (1 - (1 - (N.NI/H.R))^T)$
For typical values.

First the probability, $p(C_1)$, of one error occurring in any one column

$$p(C_1) = p(\text{active location})^{NI}$$

$$= p(a1)^{NI}$$

=> Probability, $\overline{p(C_1)}$, of an error not occurring

$$\overline{p(C_1)} = 1 - p(a1)^{NI}$$

Probability, $\overline{p(C_{1-H})}$ of this happening in all columns

$$\overline{p(C_{1-H})} = (1 - p(a1)^{NI})^{H}$$

Probability, $p(C_{1-H})$, of an error in any column

$$p(C_{1-H}) = (1 - (1 - p(a1)^{NI})^{H}$$

$$= 1 - (1 - (1 - (N.NI/H.R)^{T})^{NI})^{H} \qquad (2)$$

Giving

$$T = \frac{\ln (1 - (1 - (1 - p(C_{1-H}))^{1/H})^{1/NI})}{\ln (1 - (N.NI/H.R))} \qquad (3)$$

Thus we now have 2 equations which can be used to calculate the probability of not recovering a perfect pattern after T teaches and, at a given probability, the number of patterns that can be stored up to the given probability of error.

The following analysis illustrates the effect of altering the various parameters in the memory on recall. The picture to class memory is far more efficient than the class to picture memory for array sizes where

the picture is of much greater size than the class. The consequence of this will be discussed below.

7.7. Performance of the `Picture to Class` Memory

The first graph A_2 illustrates how the probability of error is effected by the number of class points in the class array. The parameters selected are within the range used in the simulation package written to explore the systems capabilities. It shows that a small number of class points produces less likellyhood of error for a given number of patterns taught. The graph also illustrates how the memory will effectively recover all the images with a very low probability of error until the number of images taught reaches a particular level whereupon the probability of error suddenly rises. This illustrates how the memory could be constructed so that the probability of error is very low under specified conditions. This error profile remains if other variables such as picture array size and picture points are varied. The results of these variations are shown in graphs B_2 and C_2.

Using equation 3 for T a specified error rate can be chosen. Then, for various values of parameters, the ability of the memory to store images can be estimated.

Graph F_1 shows how many patterns can be stored up to a probability of error of 0.001 whilst altering the number of points in each class pattern. The three plots are for different class array sizes. These graphs show that the class points should be as low as possible to obtain optimum storage capacity. It is worthwhile pointing out here that the number of class points should not be too small else the number of individual class patterns that can be generated would be

less than the storage capacity of the memory as an associative memory. The number of patterns that can be generated for class points of N and a class array size of H is

$$C_N^H = \frac{H!}{R!(H - R)!} = \text{Number of class patterns}$$

It will be noted that for large class array sizes the number of class points needed to obtain enough individual class patterns can be quite small. For example, the graph F_1 shows that for a class array size of 128 points the storage capacity is approximately 2000 patterns with 3 class points set per pattern. Using the above equation this gives the number of class patterns possible as C_3^{128}, which is quite adequate.

The final graphs for P -> C memory illustrates the relationship between class size and the number of images that the memory can store (Graph H_1 and H_2). Since the the class array size directly effects the amount of memory used the x axis of the graph is described in terms of memory bytes used and is equal to the class array size x picture array size. For this graph the number of class point was maintained at 5.

The important aspect of this plot is that as class array size increases the storage capacity of the system rises linearly.

The 3 plots show the storage ability for various sizes of picture array. The number of picture points is calculated for the regularly mapped edge detector input system (see later chapters). This relates to a single resolution field, broken up with edge detectors and then mapped on to the input array of the memory as shown in Fig 7.5. The nature of processing between each stage is described in chapter 9. In the example, the region covered by each edge detector is 8 x 8 pixels on I, with a reduction of 1 in 8 between image and memory.

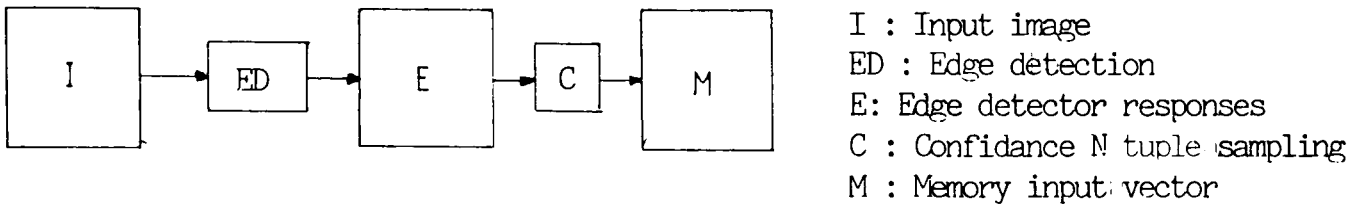


Fig 7.5
Input processing stages.

7.8. Storage Efficiency of the Associative Memory

Graphs H_1 and H_2 can be used to discover the memory's efficiency in storing images compared to a conventional file system. Each image in a linear file system is stored separately, hence the storage would equal image size times number of images to store. Similarly, the storage amount needed by the memory used here can be read off the graphs H_1 and H_2. Dividing the memory needed by a conventional system by the memory needed by this system yields an efficiency figure. Table A (in appendix) shows results for various values.

Although an exact comparison with a conventional file store cannot be made since the associative memory assumes an error level of one in one thousand images recovered, the comparisons given in the table give some idea of storage efficiency. The efficiency is greater than one in all cases, indicating better storage abilities than the conventional file store. The greatest storage improvement factor is 21.8 (21.8 times less storage needed in the associative memory compared to the listing memory), shown by the random mapped system, this yields a difference in storage ability between this system and a conventional file store of 668 megabytes to store 21500 images(total storage is 700 megabytes for the listing memory - 32 megabytes in the associative memory). This is a vast storage saving, and greatly alleviates the storage problem. It is important to note that the picture size quoted is the input to memory size, and not the actual image size.

7.9. Performance of the `Class to Picture` Memory

This memory differs from the picture to class memory in that its input array is much smaller than the output array. Also, the class array is the input to the memory, alteration of which is used to control the storage abilities of the memory. These two factors lead to quite different storage properties between the P -> C and C -> P memories.

The same equations that were used in the P -> C memory can be used here by changing the meanings of the parameters of the equations. The new assignments are,

H : Size of picture array

N : Number of elements at 1 in each picture array

R : Size of class array

NI : Number of points in each class array set to 1

As in the P -> C memory, graphs were generated using the equations given previously to ascertain the storage properties of the memory. The first graph E_2 indicates how many patterns can be stored in the memory before the probability of an incorrect recovery rises above a set level. In this graph the number of class points in each class pattern was varied, whilst holding all other parameters steady.

The graph shows a very important feature of the C -> P memory; the storage ability does not rise linearly as the number of class points is increased. The graph gives optimal storage at a particular number of class points, indicating that the number of class points need to be carefully set using the equations given above to obtain optimal storage capacity at a given error rate.

Graph F_2 shows a set of graphs of different class array sizes, maintaining the probability of error at the same level for all graphs and

varying the number of class points as in the previous graph. Again the storage ability is plotted for different expected error rates. The graph shows the same function as before, but the important factor is that the maximum storage ability occurs at approximately 18 class points in all plots (line a in graph F_2). This is an important finding. Graph G_1 examines this for a wide range of class array sizes. In this graph the maximum storage ability of the memory is plotted over a set of class array sizes. (i.e. the maximum points of the F_2 graph for different sizes of class array). The plot shows that there is a linear relationship between storage ability and class size for different probabilities of error. This is a very interesting outcome, the effect of which is to make memory design very simple. In general, for a given input image size and number of image points active in each image the number of class point can be estimated and fixed. The class array need then only be altered to set the storage level or error rate required.

Graph I_1 shows a plot of class points needed for maximum storage ability. This plot appears to be linear, as expected from graph G_1, but a detailed analysis indicates this is not the case (see graph I_2). The number of class points needed varies very slightly and so, to achieve maximum storage, the number of class points only needs to be varied slightly for different sizes of picture array etc. But since the variation is only very slight the effect is negligible when brought up to the nearest integer.

It is worthwhile asking why the relationship shown in F_2 is non linear and what causes maximisation. As the number of class points increases so does the rate of saturation of the memory, this is shown in graph J_1. But as the number of class points increases the probability of getting an error on the output decreases (for a fixed

saturation rate). This follows from equation (3) above, i.e

$$p(\text{error}) = 1 - (1 - p(\text{active location})^{NI})^{H}$$

p(active location) is a constant at a particular saturation level of memory. NI is the number of class points, as this increases p(error), decreases. From this we can deduce that,

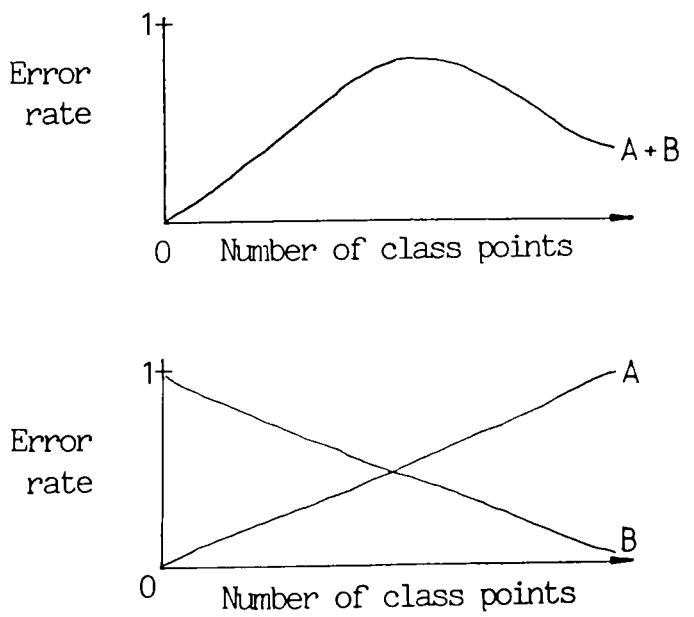
a) Increasing NI => increasing saturation rate => increasing error as NI increases.

but,

b) Increasing NI => increases power to prevent error => decreasing error rate as NI increases.

This can be plotted as shown in Graph 7.2 (below). The two individual effects combine to produce the overall effect found in the previous plots.

The C->P memory shows a lower storage capacity compared with the P -> C memory. Table B (in appendix) contrasts storage capacities in the P->C and C->P memories with equal sized picture arrays and picture



Graph 7.2
Explanation of Error rates.

points and with a set number of class points. The table shows two sets of calculations. The first (A) is the storage ability and efficiency for a picture array of 896 bits and picture points set to 64; the second (B) is for 16 picture points in the same sized array.

For results in row A, the table shows that the P->C memory stores two times the number of images than the C->P memory. Also, an important point here is that the efficient of the C->P memory under the conditions selected is less than 1, which means it is inefficient compared to a conventional file store.

The only way the C->P memory efficiency can be increased (and thus match its abilities with that of the P->C memory) is by either increasing the picture array size, or by reducing the number of picture points. Altering the class size will have no effect since it will affect both the P->C memory and C->P memory equally.

The second set of calculations (B) in the table show the effect of reducing the number of picture points in each pattern from 64 to 16. This has the effect of increasing the efficiency of the C->P memory by a large margin, whilst only increasing the efficiency of the P->C memory slightly, thus making the efficiencies of the two memories match better. This is due to the non linear change in storage abilitys (different for the two memories) as the number of picture points is altered (see previous graphs).

7.10. Matching the Storage Properties of the Memories

Formulae derived to predict the storage abilities of the memories do not take into account the effect of noise in the input pattern. If the input pattern is distorted due to noise or shape distortion, the recall efficiency of the memory will be reduced. Because the C->P

memory is expected to receive the pattern presented in teaching (i.e the N point class, and so not distorted), the formulae can be used directly to derive the storage ability and error rates required for this memory. However, the P->C memory receives the input scene and thus will not be presented with the pattern taught on in every test. Thus the error rates and the storage abilities given by the formulae will be greater than those found in practice. A consideration of recall in the presence of a noisy input image is given later.

In the above discussion it was shown how the two memories differ in their predicted storage abilities. Under certain conditions (see table B), the P->C memory exhibits far better storage properties than the C->P memory. In the light of the above this is not a disadvantage since in practice the storage ability as predicted from the formulae needs to give the P->C memory far better storage abilities than the C->P memory. Thus a system can be designed in which the differential in storage between the two memories is highly biased in favour of the P->C memory. A large bias will mean a reduced storage ability and a high resilience to noise, a low bias will provide high storage ability but a low resilience to noise

7.11. Analysis of Recall Abilities in the Presence of Noise

This section covers an extension of the basic recall analysis to see what factors are involved in recall on an input pattern containing noise. The analysis is on the same lines as above, with a more detailed look at the threshold process.

Variables

Picture array size	R
Number of elements at 1 in teach picture array	NI

Class array size H

No of elements set to 1 in each class pattern N

No of elements at 1 in test picture X

(where X < NI)

No of elements at 1 in test picture that were not a part of the original pattern S

Again we assume a memory matrix of binary points of size R x H. On each teach NI x N points are accessed in the matrix and set to logical 1. The matrix is initially assumed to be set to all 0.

As before, we need to know the probability of obtaining an incorrect recall when a given number of patterns have been taught in to the memory, where recall is the case when the recalled pattern and taught pattern differ, or have a hamming distance of greater than or equal to 1. As in the preceding analysis, it is evident that the conditions of failure differ for the P->C memory and the C->P memory. In the following analysis only the P->C memory will be considered, since it is expected that only this memory will receive a pattern that is different to that taught.

7.11.1. Recall Error for a Complete Pattern

An error occurs when, after teaching on a complete input pattern, the class pattern differs from that taught. The response of each element in the recovered class pattern will be maximal and equal to NI (the number of elements at 1 in the input pattern). An error will occur if other points in the class array have a response equal to NI. This will be due entirely to saturation of the memory (note we are considering the case where the input image is exactly that taught). The probability of one of these extra points reaching the value NI is given by NI^P

where P is the probability of finding an element in the matrix at logical 1, after T patterns have been taught.

7.11.2. Recall error in the case of an input pattern with elements set to 1 removed

In earlier chapters it was shown how the memory will fail when noise is added to the input pattern. The quantitative effect this has on the class response will now be described more fully.

To make explanation clearer, the class elements which are correctly recovered will be defined as C and the points that are incorrectly recovered defined as ~C, such that ~C = 0 and C = N if the class is correctly recovered. In the case of a complete input pattern, errors occur in such a way as to cause C to remain the same and ~C to increase. This can be shown graphically as in Fig 7.6. In this graph, all the responses X are the responses relating to the correct class pattern. The responses Y and Y´ are due to random interference within the recall process. The mean response of sum Y + sum Y´ is approximately NI^P. If N point thresholding is applied to the above to recall the class, where N = 3, all the bars in the graph marked X and Y` will be set to 1 (a total of 4 points set to 1 in the class). Thus, after recall, it will be evident that ~C = 1 and C = 3. In this case the N

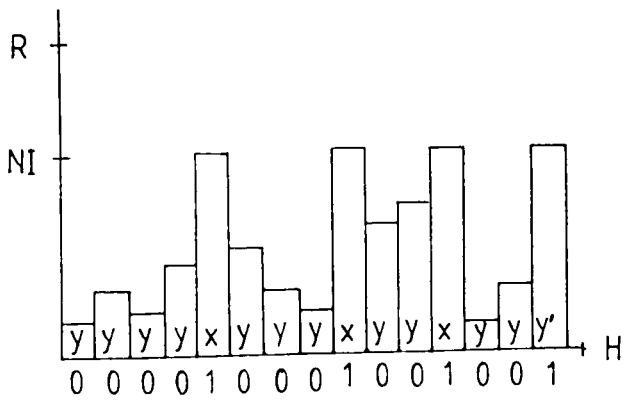


Fig 7.6
Typical response of the class output for a test on a complete pattern, where the memory is taught to saturation.

point thresholding can be reduced to a simple `effective` threshold at NI because the input pattern is exactly that taught. This effective threshold is the same as is applied to the C->P output response to recover the picture.

Now consider what happens to the graph above if the input picture taught has elements set to 1 removed, and the image is then retested. The number of elements at 1 in this pattern is given as X and is a subset of the original elements set to 1 (see Fig 7.7).

It can be seen that since only X points are now at one in the input pattern, the effective threshold has now reduced to a value X. Because of the reduced number of input points the effect of the Y responses in corrupting the recovered class has increased. (It will also be noted that the Y responses have also decreased in comparison with the original).

7.11.3. Effect of N Tuple Sampling on Recall

We must now incorporate the effect of N tuple sampling (where N pixels are associated together using a logical operator) the input image before presentation to the memory. In the case above N tuple sampling was assumed not to occur, the input pattern was fed directly to the

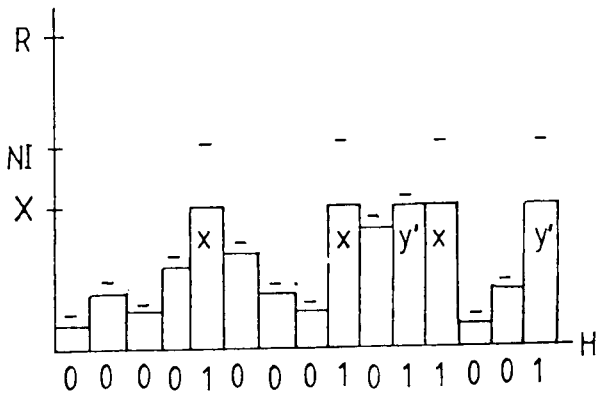


Fig 7.7
Response of an input pattern where elements at 1 are a subset of the original pattern taught.
(`-` indicates the level associated with a complete input pattern)

memory. For the analysis of a complete pattern there is no change in the result because the input to the memory has not changed in testing compared to teaching. In the case where the input pattern is reduced the only effect is to accelerate the reduction in X as the pattern subset is reduced, and so the conclusions above still hold.

7.11.4. Effect of Noise Peripheral to the Pattern on Recall

In considering the effect of random noise on the system we must take into account the N tuple process acting on the input field. We consider here the random mapped type N tuple sampling with no all 1's or all 0's addresses (This will be explained in chapter 9, basically theses N tuple states do not contribute to the recognition of the pattern). If we consider a simple pattern presented to the system, and then the system is tested on this pattern with varying amounts of noise added, as shown in Fig 7.8. With no noise added the number of elements at 1 in the array presented to the memory will be X, as noise is added a further S elements will be included in this pattern up to a maximum of, X + S = R'/N where R' is the input image size (before N tuple sampling) and N is the N tuple size. If a large amount of noise is added all 1's addresses will be generated reducing X + S eventually to zero (i.e an empty array being presented to the memory). This is shown in Fig 7.9.

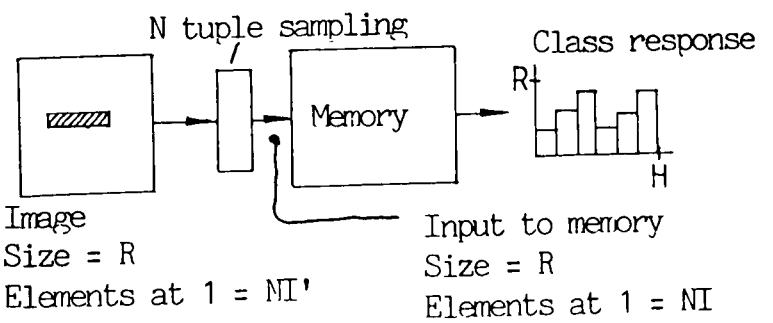


Fig 7.8
A simple pattern presented to the system.

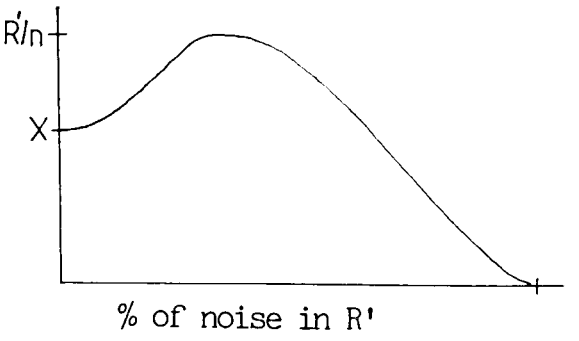


Fig 7.9
Change in class response as noise is added to the input pattern.

So as noise is added both N tuples activated by the original pattern and N tuples not previously active will be effected (active: meaning registering a tuple which is not all zero or all 1). The graphs in Fig 7.10 and 7.11 show the typical changes in the response of the class to a taught complete input picture, before and after noise is added. The original correct class pattern V is both decreased due to 'correct' N tuples on the input and increased due to other N tuples not in the original pattern becoming active. The elements in the class pattern not in the original taught pattern also change.

The processes involved here are complex and would be the topic of

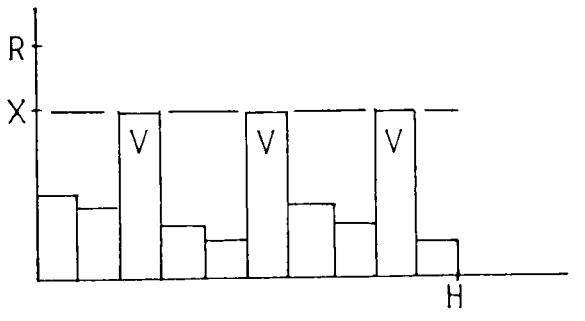


Fig 7.10
Class response - no noise added.

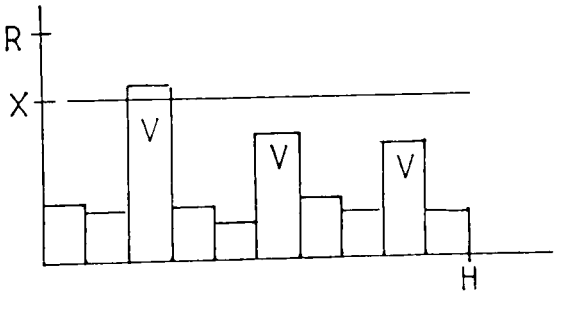


Fig 7.11
Class response - with noise added.

further studies. It is true to say that it is impossible to predict accurately the effect of 'real' noise (i.e non random) on the memory as opposed to the ideal case (as studied above) where noise is purely random. Further complications need also to be investigated such as the effect of using local N tuples formed out of edge detectors and the effect of having different patterns taught under the same class pattern as is done in the final scene analysis system.

7.12. Processing the Input Image to Allow Memory Storage Control

The number of picture elements set to 1 in the input image array must be restricted in some way if storage control of the type outlined above is to be allowed. Chapter 9 discuses the use of two different input processing systems, random mapped and confidance tupling. In the process of random mapping control can be in the form of altering the N tuple size. In the confidance process edge detectors are used and a similar control can be also be used here. In some cases it may be necessary to have a different number of picture elements set to 1 in the array feeding the P->C memory and the array feeding the C->P memory (i.e to obtain the desired storage properties). This can be done by randomly removing N tuples in one input array until, say, only 16 N tuples are 'active'. This process has not been used in practice and remains theoretical.

7.13. Summary

The above discussion illustrated the general properties of the memories by considering their probabilistic behaviour. It can be seen that the P->C memory has very good storage abilities compared with a conventional linear file store. The C->P memory does not have such efficient storage but, since it receives undistorted patterns, it can

be used effectively. In general the memories can be designed to fit specifications, although further work is needed, especially in predicting the recall abilities in the presence of noise in the input image.

It is now possible to analyse the memory`s performance experimentally with use of simulations of hardware implementations. This will be covered in the next chapter.

CHAPTER 8

Empirical Investigations into Memory Storage

8. Introduction

The following covers the experiments done on a computer simulation of the memory structure laid out in chapter 6. The theoretical behaviour has been considered in chapter 6. The practical performance will now be examined and the correlation of the predicted and the experimental behaviour assessed.

It will be shown that the results differ for the predicted onset of recall failure. The mathematical probability of failure was given as 'the probability that perfect recovery of a pattern will not occur after T patterns have been taught into the memory'. The experimental data was derived from the following 'The number of patterns that can be taught before the Hamming distance between a taught and tested pattern, averaged over a number of tests, becomes greater than one' i.e. the point at which the memory fails to recall correctly after a number of patterns have been taught. A formula for calculating this latter condition has been derived from the existing formula (see appendix 2 for derivation). This is given as a function of H,NI,N and R as defined previously and is :-

$$T = \frac{\ln\left(1 - \left(1/H^{(1/NI)}\right)\right)}{\ln\left(1 - (N.NI/H.R)\right)}$$

H = No of elements in the column vector.
NI = No of points set to 1 in the row vector.
N = No of points set to 1 in the column vector.
R = Size of the row vector.
T = No of patterns taught before hamming distance of 1.

8.1. Simulation Details

Simulations were performed for the basic associative memory as described in chapter 5, the basic form of the memory is shown in Fig 8.1. The picture vector was fed directly into the rows of the memory matrix; similarly the class vector fed directly onto the columns. The data used for the picture vector was 'binary simulated tupling', intended to to give a close approximation to the data that would be produced by the regularly N tupled confidance system in chapter 9. The input picture vector was broken into N tuple groups, each group only having one bit set at random within it (see Fig 8.2).

It is useful to explain the process of confidance tupling which is defined in chapter 9. First the input scene is first edge processed. Four edge operators of different orientations are used to form the

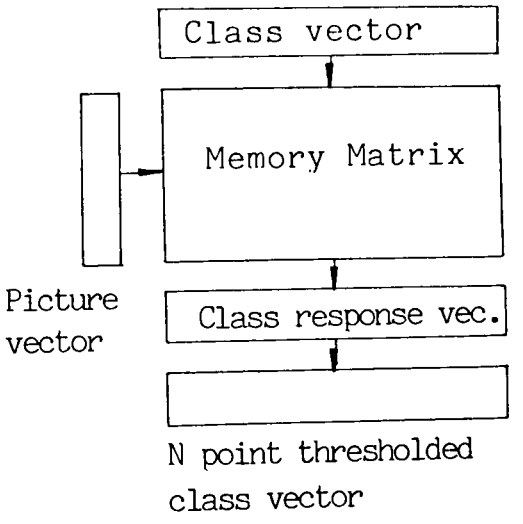


Fig 8.1
Basic simulation model for the picture to class memory

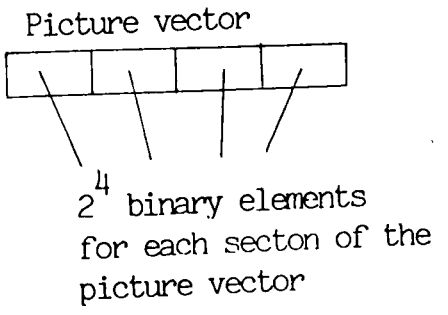


Fig 8.2
Construction of a binarised picture vector.

data for one N tuple. The values from the 4 edge operators are then reduced to one of 16 states by the process of confidance tupling.

8.2. Picture to Class Simulations

The results of these simulations are given in the graphs Y1 to Y3 in the appendix. Their purpose is primarily to substantiate the equations derived previously.

The first three graphs show plots of mathematically predicted and simulated results for experiments where the number of points set to 1 in the class array varies. Each graph has all other variables constant, with differing class array sizes for each.

Table of variable settings.

Graph	Picture Size	Picture Points	Class Size	class Points
Y1	896	22	32	1 - 30
Y2	896	22	64	1 - 64
Y3	896	22	96	1 - 96

The procedure used for all the graphs was as follows,

1> Teach N patterns into the memory. On each teach store the class pattern and picture pattern used to teach each picture. Randomly generating both the class pattern and the binary simulated tupled image.

2> Test on first N patterns in teach set. After each test calculate the Hamming distance between the stored class and the recoved class pattern, then average the Hamming distance for all tests.

3> If the average Hamming distance is greater than 1, then stop, else

teach a further P new patterns.

The reason for testing only the first N patterns taught on each cycle is due to the problem of storing a vast number of patterns. In all cases N was set to 20. The coarseness of the graphs was due to the size of P (the number of new patters taught on each cycle), this was set to 100 in all cases. This provided a programme that ran in an acceptable length of time (many days in most cases).

It can be seen that the predicted results show a good relation to the experimental results (shown best by graph Y3), thus supporting the mathematical analysis.

8.3. Class to Picture Simulations

These graphs show results from experiments with the class to picture memory (recover picture from class). Two sets are provided, Z1 - Z3 and Z1.1 to Z3.3 . These results are slightly more complex to interpret than for the P -> C memory.

Graphs Z1 to Z3 show the number of images that can be stored in the memory for an optimal number of class elements at 1 and for each class array size. The number of picture elements at 1 and the size of the picture array was held constant for each plot. The picture image was formed in the same way as previously i.e. a simulated N tuple image was used. Basically, for each class array size, the number of class points needed for optimal storage was found. The plots show the class size and the number of patterns taught. Recall was said to fail when the picture recalled differed from the picture expected by one bit - averaged over a number tests in the same way as in the previous experiments.

In each graph the number of picture points set to 1 was varied, this is shown in the following table,

Graph	Pic. array size	Pic. points	Class array size
Z1	896	22	1 - 800
Z2	896	16	1 - 400
Z3	896	13	1 - 800

The results are compared with those in graph G_1 in the previous mathematical predictions.

The results compare favourably with the expected results, although they are a little lower than expected. The reason for the difference is not clear but is probably due to approximations that were entered in the equations.

The second set of results Z1.1 Z2.2 and Z3.3 are coupled to the previous graphs. These give the number of class points needed to obtain maximum storage. They are paired to the previous graphs in the following way, Z1 with Z1.1, Z2 with Z2.2 and Z3 with Z3.3.

These graphs show a wide variation in response, although it can be generaly concluded that the number of class points does not rise as the class array size grows. The reason why the results are not smooth is because the number of patterns tested after each teach cycle was perhaps too small and the number of class points needed is obviously very sensitive to the randomness present in the teach patterns. Over all, the results do not contradict what is expected, i.e that the number of class points needed for optimal storage in the C->P memory remains constant over a large range of class sizes.

8.4. Summary

This chapter has compared the predicted storage properties of the associative memory described in chapter 6 with experimentally derived results, the equations used for comparison were derived from the equations in chapter 7. In general the results show that these equations can be used effectively to predict the storage abilities of the memory in the case where the input pattern is complete.

CHAPTER 9

Optimisation of N Tuple Samples

9. Introduction

In the N tuple process described in chapter 3 various optimisation schemes for the mapping of N tuple samples within the input image were described. This chapter covers the specific mapping strategies for reliable recognition of edge representations of occluded objects.

9.1. Occlusion Analysis and Random N Tuple Sampling

As explained in chapter 4, the input image needs to be edge processed to allow occlusion analysis to be effective. This results in problems when using random N tuple processing.

Consider a large image containing a small shape as shown in Fig 9.1. Because only a small amount of the image is filled with black lines only a small amount of the input array will contain elements at logical 1.

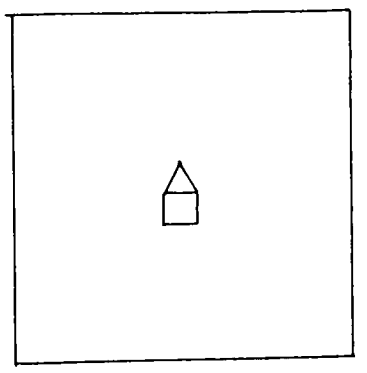


Fig 9.1
Small line drawing in a large scene

If an image is considered convolved with N tuple samples from a set of decoders at random, it is likely that only a few N tuples will have decoder lines which fall on an object edge (a dark area of the image).

The probability of obtaining an all 0's N tuple state when an n tuple sample is randomly placed within a scene is given by,

$$P_i = (1 - H/N)^n$$

For N >> n

H = Number of dark pixels

N = Number of light pixels

n = Tuple sample size

This gives a probability of 0.96 with H = 10, N = 1000 and n = 4, This means that for every pattern taught most of the N tuple states will be all zero. If the input image is taught at various positions within the scene, most of the locations in the memory matrix concerned with storing the 2^0 N tuple states will be set to 1.

After a number of patterns have been taught the locations in the memory which record the occurrence of all 0's N tuple samples will be filled or `saturated`, thus they will have no discriminating power and, as a consequence, have no part to play in discriminating the patterns. This process also occurs to a lesser extent in line drawn images of larger shapes. A problem exists during recall when these memory locations are near saturation. The responses from the all zero N tuples will mask the responses from all other samples (defined as 'active' tuples), expressed as a large amount of `background` noise in the recalled class pattern.

Because of this problem the locations for all zeros detector lines are not implemented in the memory. This overcomes the problem above and introduces a useful property into the system. This process is equivalent to training on an all zeros input pattern as investigated by Wilkie (Wilkie1983).

Now that all zero locations are not represented in the memory, the average response from the memory will depend on the size of the object being recognised or, more accurately, the number of elements at logical one in the object. Consider the two simple patterns shown in Fig 9.2. Pattern A has more black lines present than pattern B, thus pattern A is likely to create more 'active' tuples than pattern B. The total number of active lines entering the picture input to the memory will therefore be greater in A than in B.

If we teach A under one class and B under another, then present both A and B in the same scene as shown in Fig 9.3, on testing the memory will 'see' A rather than B. This is because A has more active N tuples than B. This would not occur if all zeros N tuples had been included, in that case A would produce the same amount of N tuples responding for its pattern as B, this would be reflected in the output in that the response for A would equal that for B, leading to confused recognition.

This is a very important feature of the system allowing the analysis of complex multi object scenes. The system would identify the major shapes in the scene first then, by the process described in chapter 4,

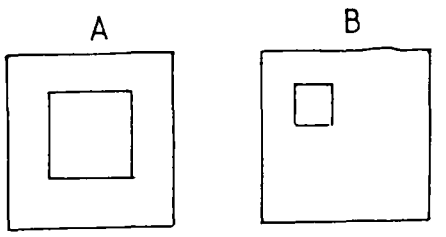


Fig 9.2
Two different sized squares.

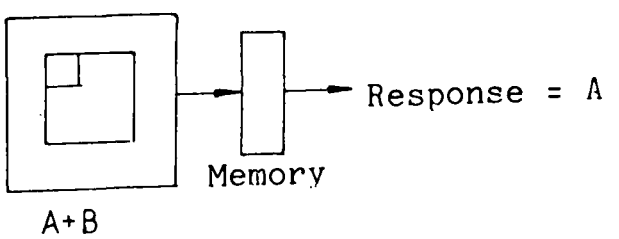


Fig 9.3
The large square is recognised before the small square

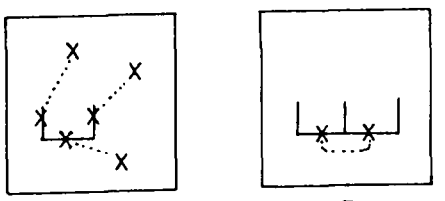
go on to iteratively find other shapes in the scene.

The fact that line drawings only activate a small percentage of the pixels in a scene causes problems by restricting tuple codes in 'active' tuples. It is quite likely that only one line from any tuple connects to an `on` pixel in the scene. Thus N tuple codes are reduced to those N tuples containing only one bit set to 1. This is a sever problem since the logical process that operates due to the random mapping of the tuples is defeated. This is illustrated in Fig 9.4. The two images need an N tuple to straddle both 'U' positions to allow recognition of both A and B to take place (Suitable positions for an N tuple are shown in the diagram). This is unlikely to happen in a random mapped system. The solution to this can only be by controlling the mapping of N tuples in some way, this will be expanded on in the next section.

9.2. Non Random N Tuple Samples

To prevent the restricted N tuple codes as described above, the mapping of decoder lines on to the image array must be restricted in some way. The N tuple samples must be distributed both globally to allow for logical relations over a large area to be recognised, and also locally for similar reasons.

Local N tuple samples need to be grouped closely to recognise local



x....x : One N tuple sample
(N = 2)

Fig 9.4
2 U`s problem

logical relations as shown in Fig 9.5. Since the area of spread of the N tuple samples is small, it is likely that a wide range of N tuple codes will appear. If the distribution area is enlarged and the N tuple size is maintained, the likelyhood of a wide range of N tuple codes is reduced. For very large distribution areas the number of codes is very small. Fig 9.6 illustrates a case where an N tuple sample is distributed globally.

To offset this process the receptive area of each N tuple line can be increased (i.e pixel size). Thus the probability that an N tuple line will see an edge increases. This is shown in Fig 9.7 for the shape shown in Fig 9.6. Note how the N tuple receives more inputs at logical 1.

In general, as N tuples are distributed over an increasingly larger

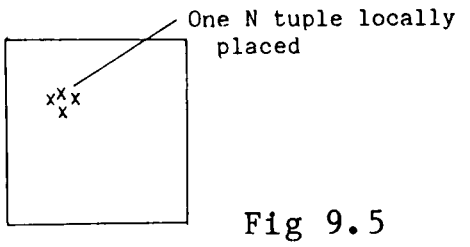
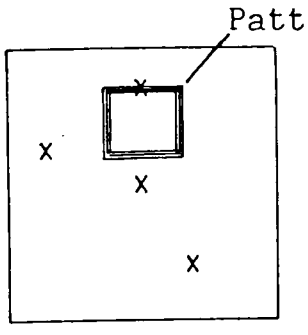


Fig 9.5
An example of locally distributed N tuple samples



x : N tuple lines

Fig 9.6
An N tuple sample distributed globally

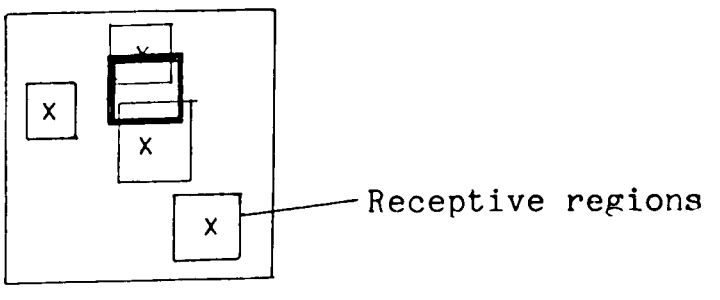


Fig 9.7
Global N tuple sampling with large receptive fields

area, the receptive fields can be made larger in sympathy. The optimal distribution of N tuple samples will depend on the patterns being taught.

The system has local N tuple samples with small receptive fields, and more globally distributed N tuple samples with larger receptive fields. This allows a wide spread of N tuple codes to appear for both local and global N tuple samples when viewing line or edge processed drawings.

Producing the distribution of N tuple samples given above was thought unnecessary in preliminary work; a simplified scheme was thought adequate at this stage. The processing used in the simulation was as follows.

The input image was windowed by a set of processing windows or fields of differing resolution. The resolution was uniform within each window. In the simulation two such windows were implemented, this is illustrated in Fig 9.8. Furthermore, the windows were labeled 'high resolution' and 'low resolution' for obvious reasons. Each window was

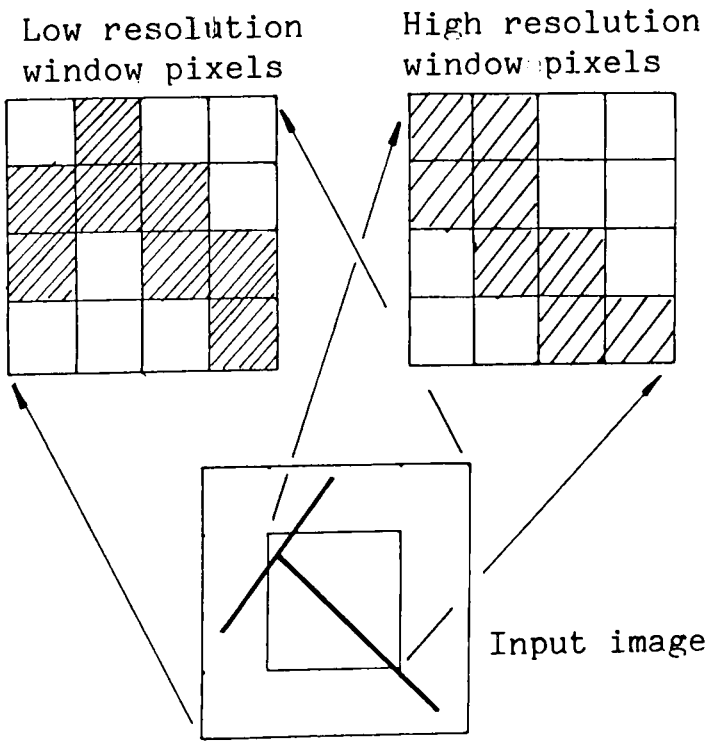


Fig 9.8
Views from high and low resolution windows

made up of an equal number of pixels, and placed concentrically with each other, thus the high resolution window only looks at the center section of the image. By reducing the centre window size in this way storage cost are reduced because the memory size is linked directly to the window size.

Because the high resolution window does not cover the whole scene, some form of camera control is necessary to center the high resolution window at areas where detailed discrimination is needed. The following chapters discuss this aspect of the system.

The concepts used above relate closely to the construction of the human visual system. It is known (Wright1983) that the retina of the eye contains a high resolution `foveal` region and a low resolution peripheral region. The reasons why such an implementation is used in the human visual system are likely to be similar to the reasons considered here in that the processing capacity is limited, although it is unknown whether any N tuple process occurs in humans.

9.3. Organisation of N Tuple Samples

Random mapping of N tuple samples over the scene causes problems when an image of an occluded object is considered. The image shown in Fig 9.9 illustrates this problem. It shows a small shape near the centre of the input image surrounded by other shapes. If we consider that the N tuple samples are randomly mapped over the scene. The global N tuple samples (ones which span the whole scene) will receive data from the shape in the centre of the scene and be effected by the other objects in the scene. The effect of this is to corrupt N tuple samples which receive inputs outside the shape to be recognised. To overcome this the N tuple samples are restricted to local regions, which

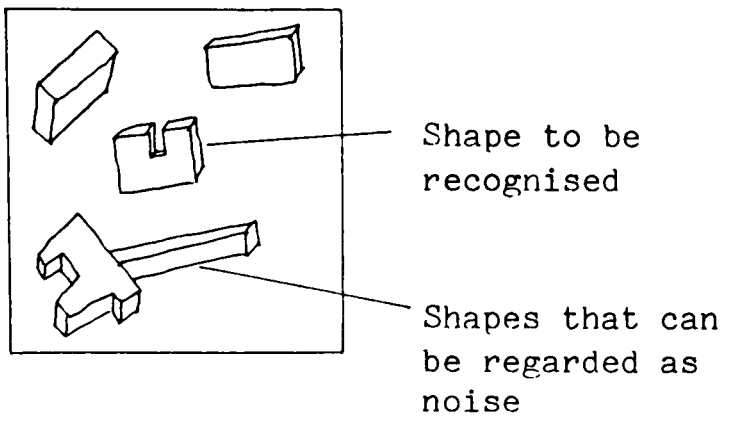


Fig 9.9
A cluttered scene

prevents N tuple samples from being corrupted by other objects in the scene. Because the receptive fields of the individual inputs to each N tuple sample in the low resolution field are large, global relations between different parts of the scene can still be recognised, although some restriction has occurred. The addition of more windows at greater resolutions overcomes these restrictions.

Fig 9.10 shows examples of a low and high resolution N tuple samples. The high resolution sample consisting of four pixels (labeled 1 to 4) is shown in B, the low resolution sample, made up of larger pixels, in A. Effectively the low resolution tuple will detect coarse features within patterns, whilst the the small high resolution tuple will identify small local features.

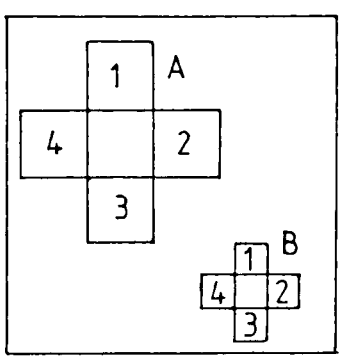


Fig 9.10
An example of low (A) and high (B) resolution N tuple samples

9.4. Edge Operator Construction

The discussion above outlined the following processes to map the input image on to the N tuples of each decoder. The image is first broken up into two fields, one large covering the whole image the other small covering a small centre section. The large field would be broken up into x regions or pixels and be used for coarse detection of information in the image. The small field would also be broken up into x regions, each smaller than the ones in the low resolution field. This would be used to detect fine detail.

The next stage is to map the values from each pixel on to the decoders. If we consider decoders with 4 inputs (thus 2^4 outputs) these would be taken from the fields in local groups.

At this stage the decoders require binary inputs, i.e. logical 1 or logical 0. Thus each pixel must signal 1 or 0 depending on the contents of the region it is assigned to.

We assume that the input image originates from an imaging device, such as a line scan television camera. It is then processed into a high resolution pixel image of grey scale values, perhaps of 256 by 256 pixels. This image is then reduced to create the low resolution window, of perhaps 32 by 32 samples, and the high resolution window also of 32 by 32 samples. The high resolution image may need no reduction to achieve the required sampling, other than selecting the center 32 by 32 pixels of the input image and mapping each pixel directly onto each sample. The low resolution image needs to reduce the input image of 256 by 256 pixels to one of 32 by 32 samples. Effectively, each sample in the low resolution window receives data from an 8 by 8 pixel region in the input window.

It was pointed out in section 1 that the image is to be edge processed. A simple form of edge operator was needed to preserve processing speed during simulation, because a vast number of operations would be needed to convolve an image with the edge operator.

The placing of the edge operators is shown in fig 9.11. This organisation was selected to provide optimal detection of vertex constructions within the scene. A four tuple arrangement is shown, with each edge operator set at 45^{o} intervals around a central point. Further edge operators can be added to increase the N tuple sample size, as each operator provides an N tuple input.

There are many designs for edge detectors (seeDavis1976,Torre1986), the choice was based on the following criteria,

1) The detector must be orientation specific.

2) They must not be upset by changes in overall illumination.

3) They must detect an edge of the preferred orientation anywhere in the detector area.

4) They must be equally good at detecting edges and lines.

5) They must not be unduly complex, i.e. must work fast.

The detector must be orientation specific because of the overlapped construction shown in Fig 9.11, i.e.detector 1 must be orientation

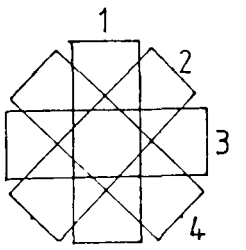


Fig 9.11
Edge detector orientations (1 - 4)

sensitive to lines and edges within a range not extending into detector 2 or 4's sensitivity region (see Fig 9.11). If this were not so, a lack of detail would result in the N tuple state produced by each group of edge detectors i.e. a vertical could be given the N tuple state for an 'X' (a vertex).

The illumination condition (2) is important, consider the graphs of two linear light intensivity profiles shown Fig 9.12. Even though the average light intensity in A and B varies, the edge strength must be registered as the same, i.e. the strength is given as the difference between the maximum and minimum light values of the defined region.

The ability for the detector to detect an edge anywhere in its receptive area is also important. To explain why, consider the edge operator in Fig 9.13 which does not forefil this criteria. This detector is modeled upon an edge detector thought to be used in the human visual system, see (Davis1976).

The detector processing is shown at the top of Fig 9.13, edge profile `A` produces a response as shown in the graph below it. Similarly for edge `B`. In both edge profiles the central position of the edge is the same but as the detector is scanned across the edge, it signals a different position for the edge in 'A' and 'B'. The result of this is to confuse recognition in some circumstances. Consider a grey circle

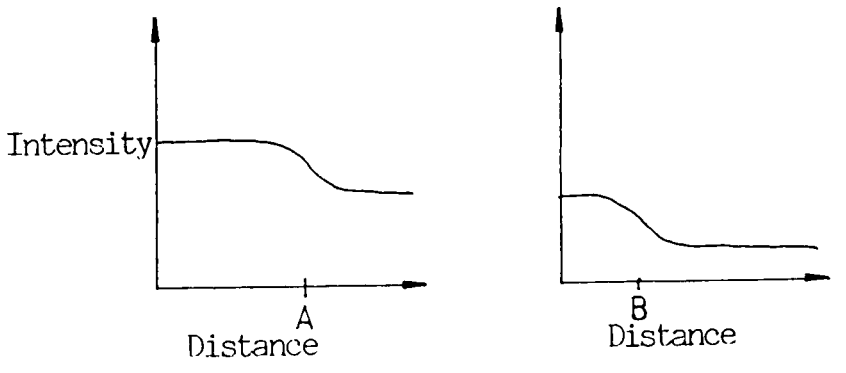


Fig 9.12
Simple light intensity profiles of an edge

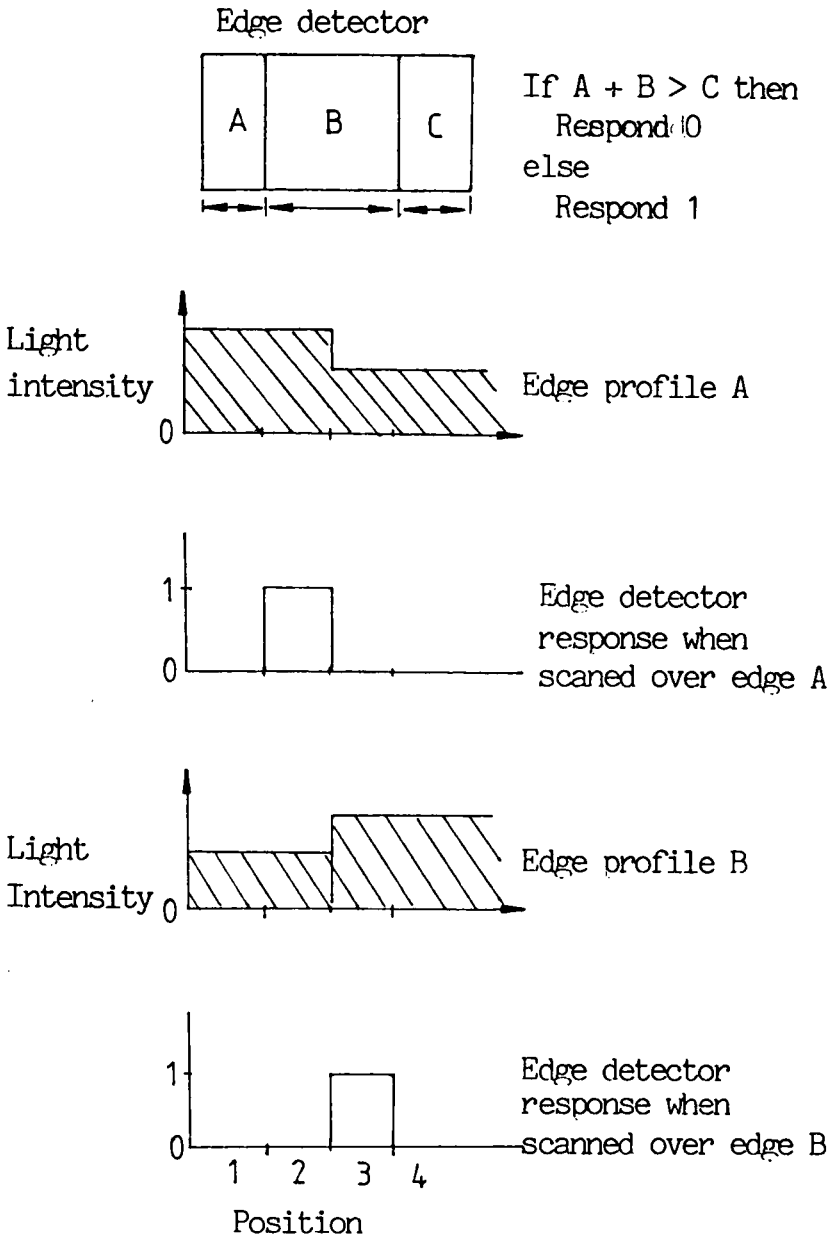


Fig 9.13
A simple edge detector

on a white background then on a black background. The result if the above edge detectors were viewing the circle would be a different sized circle in each case. This would mean that a system trained on a circle on a black background might not be able to recognised the same circle if placed on a white background, unless it receives experience of this situation.

To overcome the above problem a simple Roberts (Davis1975) edge detector shown in Fig 9.14 was used. This detector fulfills all the requirements listed above, It is easy to implement and fast to compute. It is not effected by overall illumination as both A - B and B - A remain constant if both A and B are varied. The detector is able to detect lines as well as edges with good selectivity, dependent on the width of the detector. The detector gives a good linear response which

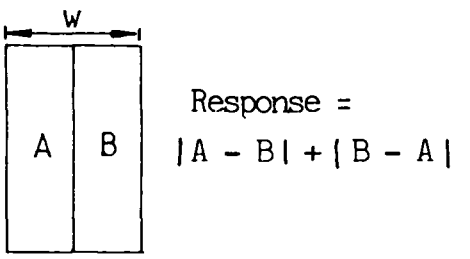


Fig 9.14
Edge detector used by the system

may be thresholded to produce a binary output.

The output response of the edge detector described above is shown in images op6 and op7. Op6a is an image of 128 x 128 pixels with 64 grey levels of a simple star pattern drawn on paper and presented to the camera. The four outputs b,c,d and e represent the responses from the 4 different angles of edge operator expressed in the form of grey levels, the whiter the area the more the detector responded to its preferred orientation. The detectors were 16 x 16 pixels square, sampled every 8,8 pixels in the image ´a´. The detectors show good selectivity to their preferred orientation.

The sensitivity to a line drawing compared to a block filled drawing is shown in op7. Both images are 128 x 128 pixels, generated as described above. The output responses from the 90 degree edge operator is shown. The output illustrates how a ´block filled´ and ´line drawing´ can be reduced to the same representation., allowing recognition of both types of images after exposure to just one.

The following input processing stages have been defined,

1) Reduction of the input image to a grey scale pixel image.

2) Calculation of edge detection values for 2 input windows of different resolution.

3) Construction of logical N tuples and decoding to one of 2^N states.

Fig 9.15 illustrates the elements of the system described so far.

9.5. Edge Operator Thresholding

The edge operators described previously need a specified threshold to convert the response to a binary value for the formation of N tuples. Whilst this process is fast it suffers from some problems.

The main problem involves the selection of a suitable threshold which covers all lighting conditions experienced in natural scenes. The simplest process is to select a theoretical threshold but this invariable means that data is lost in very bright or very dark areas of the scene.

A better way is to derive the threshold from the scene data itself. To do this the average response of all edge operators can be taken and used as a threshold. This process is acceptable since it would allow a threshold that varies in sympathy with the overall lighting levels in

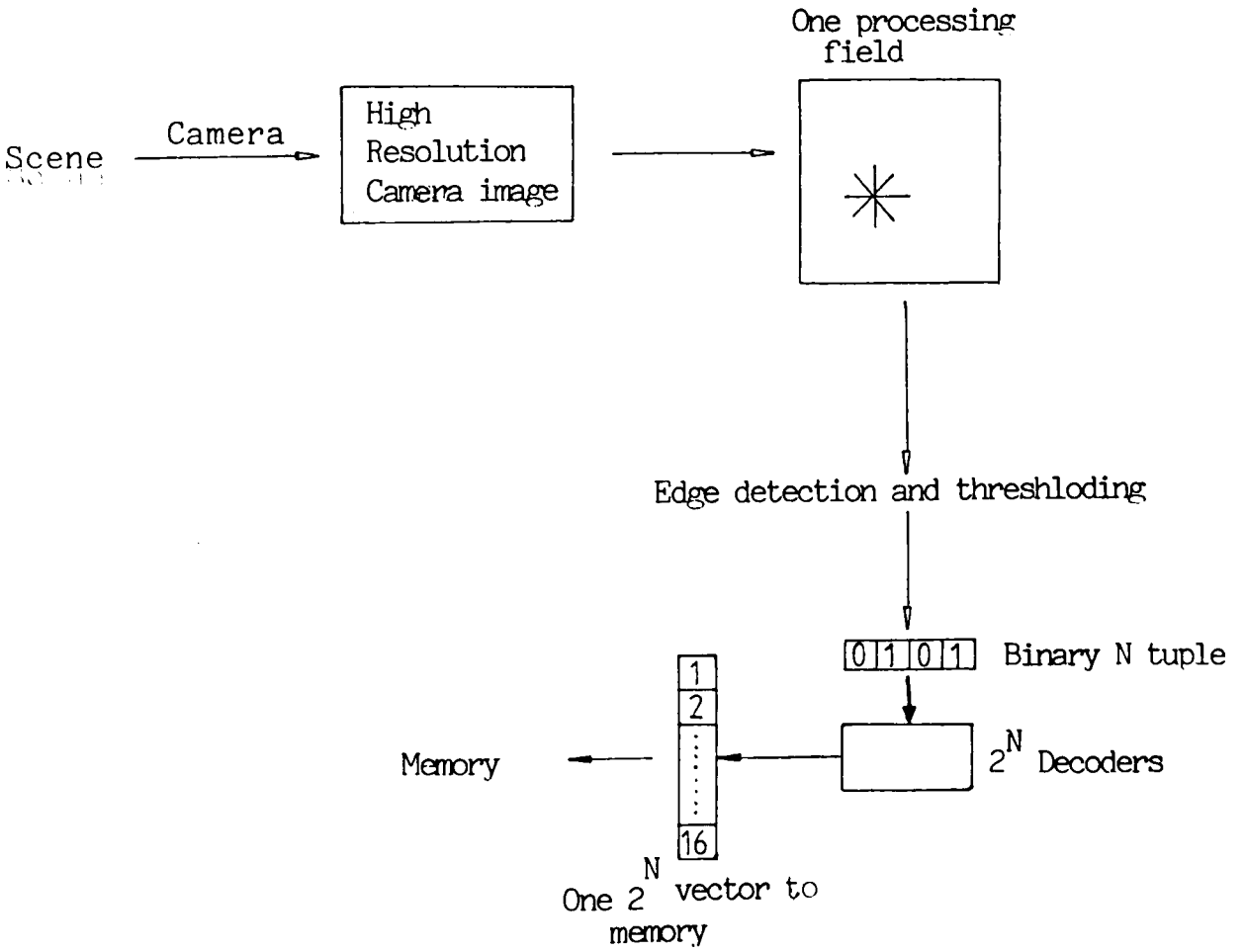


Fig 9.15

the scene.

Problems still occur with this approach when one considers a large scene containing brightly lit and well defined edges in one region of the scene, and dimly lit and less well defined areas in other regions of the scene. Fig 9.16 shows such a situation. Two N tuple groups are considered. N tuple group (i) is placed in a well lit part of the scene such that the responses from the N tuples are high. The tuples in group (ii) are placed in a less well lit part of the scene. The average response of all edge detectors are shown, along with the N tuple codes generated after thresholding. If the tuple codes indicated are compared with the profile of the responses of the edge operators, there is little similarity. What is evident is that the information contained in the tuple responses is not expressed well in this threshold process.

A process was needed which faithfully represented the response profiles of the N tuple samples and which is insensitive to local and global average changes in light intensity.

After consideration of what remains invariant over all light levels the following solution was used.

If a single N tuple group is considered the ordering of the responses

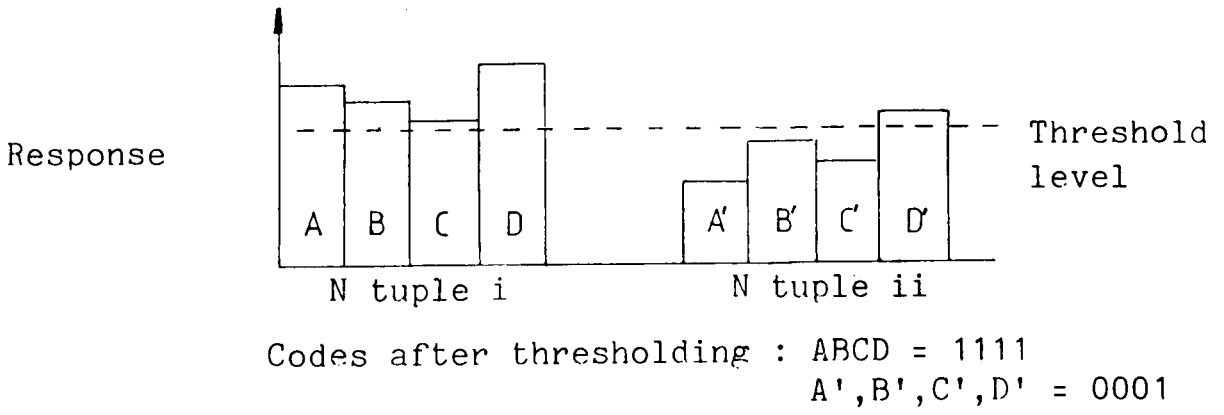


Fig 9.16
Responses from the two sets of edge detectors

from the operators making up the N tuple remains constant. If figure 9.16 is considered, the ordering of the responses A,B,C,D and A`,B`,C`,D` is the most important factor in the data to be expressed.

Consider N tuple group (ii), the ordering of this group is D,B,C,A with D maximum response and A minimum. If this ordering is mapped back into the scene the relevance of the process becomes obvious. This is shown in Fig 9.18 (the labelling of edge detectors given in Fig 9.17). The responses given in Fig 9.18 relate to the strength of this edge as ´seen´ by each detector.

This coding procedure allows the grey scale `landscape` within the detector region to be recorded. It is insensitive to overall lighting levels both locally and globally.

This process is implemented in the following way. The number of distinct orderings of N edge detectors under this scheme is $N^N-(N-1)^N$, which, for the example above (N = 4), gives 256 orderings of the responses- which is much more than the 2^4 states available in the normal N tuple process. If all 256 states for each N tuple sample were to be recorded the memory used by the system would be very large.

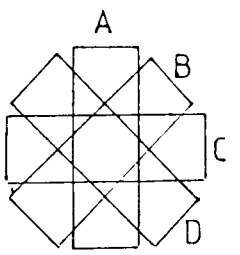


Fig 9.17
Edge detector labeling

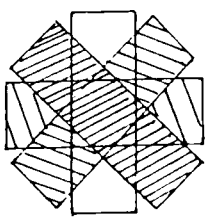


Fig 9.18
An N tuple ranking mapped back into the image domain

It is important to note that this value (256) includes the instances where all the responses are equal or cases where pairs of edge detector responses are equal. The occurrence of these states would be susceptible to any slight deviations in the input pattern, which is undesirable. The process can be made insensitive to these small changes by reducing the number of states each N tuple sample can detect.

Fig 9.19 illustrates the process. The first stage is to find the maximum and minimum responding edge detectors within one N tuple, the range of responses between these maximum and minimum responses is then broken up in to regions. In this case two regions are used, one for responses determined as being within the the maximum responding range, the other for responses lying within the minimum responding range (1´st and 2´nd regions in the diagram). The overlap region is the case where it is not clear whether the response is to fall in the 1´st or 2´nd region. For the edge detector responses shown in Fig 9.19 the ranking is C and D in the 1´st rank, A and B in the second rank. In the simulation of this the intermediate region was not implemented. If this was included it would produce two possible states if one edge detector response fell within the intermediate range, i.e if B fell into this region instead of into the lower rank, the states signaled by the N tuple sample would be,

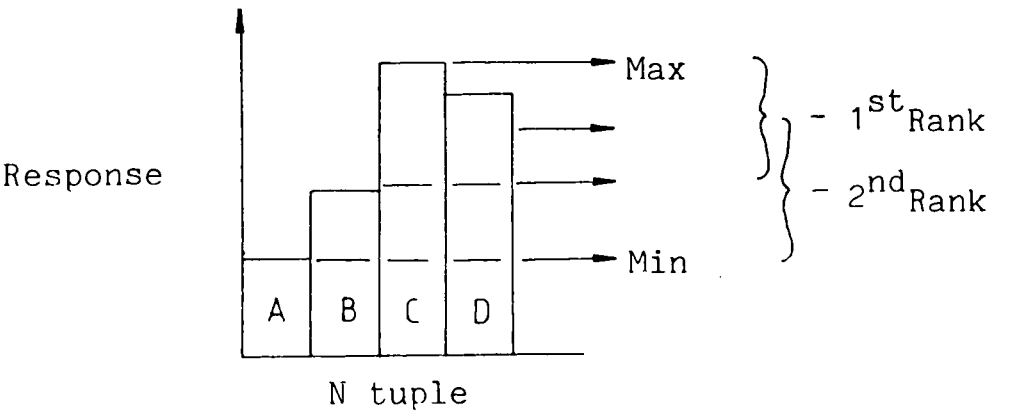


Fig 9.19
Assignment of thresholds

	1´st rank	2´nd rank
	B C D	A
or	CD	AB

In general, the number of possible states with this process is given by 2^N where N is the number of edge detectors within one group (synonymous with an N tuple).

Whilst this approach may seem like a local threshold scheme, with ordering of no particular importance, the theory allows extension to a more varied number of states. For instance if there were 3 threshold regions, the number of possible states becomes $3^N - 2^N$. This case is shown in Fig 9.20. The possible states in general equals $R^N - (R-1)^N$ where R is the number of threshold regions or ranks, and N is the number of edge detectors (N tuple size).

The number of possible states and codes used for a two ranked system is shown in table 9.1 below.

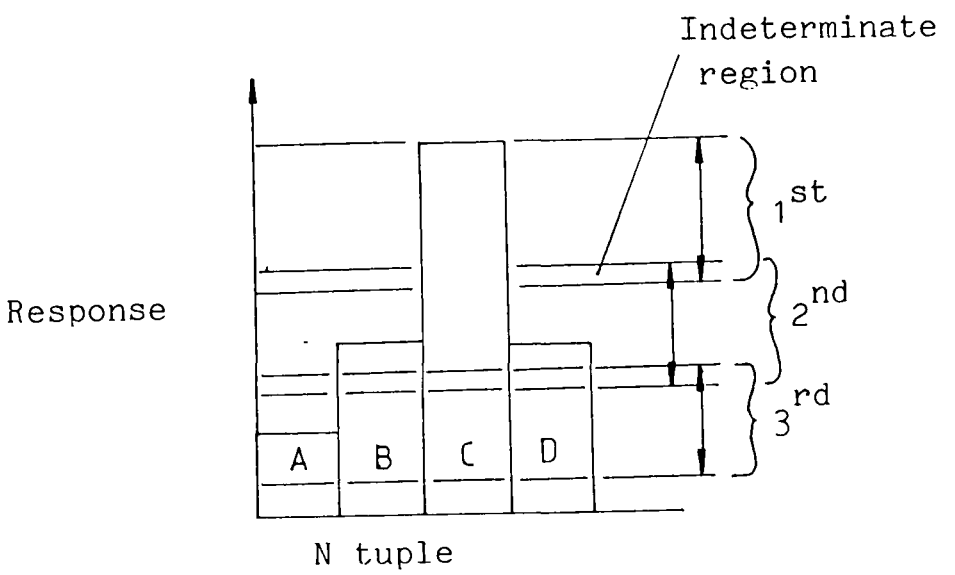


Fig 9.20
A three ranked example

RANK		TUPLE CODE
1	2	ABCD
A	BCD	1000
B	ACD	0100
AB	CD	1100
C	ABD	0010
AC	BD	1010
BC	AD	0110
ABC	D	1110
D	ABC	0001
AD	BC	1001
BD	AC	0101
ABD	C	1101
CD	AB	0011
ACD	B	1011
BCD	A	0111
ABCD	-	1111 *
-	ABCD	0000 *

* = See discussion

Table 9.1
N tuple codes generated for different
rankings of edge detectors

This table can be converted to the normal tuple representation of the 16 binary numbers, which can then be used as the inputs to the 1 in N decoders used in the N tuple process, and then applied to the memory in the normal way. The process of converting each particular ranking to a binary number is simple. If we assign one bit of a 4 bit code to each edge detector, i.e bit positions of a 4 bit code are A,B, and C, then A represents the edge detector at 0 degrees, B for the detector at 45 degrees, C for 90 deg. and D for 135 deg. We then set the bit to 1 if the edge detector response falls in the first rank, and set it to 0 if it falls in the second rank. The tuple codes in table 9.1 were formed by this method.

It can be seen that a problem occurs when all edge detector responses are equal. In this case ranking is either 1´st = ABCD and 2´nd = none, or 1´st = none 2´nd = ABCD. This case must be covered in more detail, as it is equivalent to the all 0´s and all 1´s tuple codes discused

earlier in the chapter. These codes are not represented in the decoder process because they frequently occur in images and cause saturation of memory locations addressed by them. The problem here is that the particular codes can be generated by two types of input pattern. If all edge detectors in a tuple see uniform grey level, then all the edge operators responses will be zero and generate the ranked codes 1111 or 0000. The problem is that this code can also be generated by a star pattern as shown in Fig 9.21. This pattern needs to be represented in the tuple coding in a different way to the pattern for an all grey input. In the arrangement described above a star pattern and an equal grey level input pattern are not assigned an N tuple state, the solution to this problem is covered later in the chapter.

9.6. Examples of Grey Level N Tuple Processing

The result of grey level N tuple processing is shown in op8 and op1 acting on a simple square and a more complex face. Op8 is a 128 x 128 grey scale image of a simple drawn square. To represent the N tuple code formed, the image of the tuple output is given in a graphic form. This shows a collection of four possible orientations of edges for each tuple. The ones shown are the edge detectors whose responses' falls in the upper rank. i.e the four edge detectors at 0, 45, 90 and 135 degrees are given the following patterns `|`, `/`, `-`, `\`

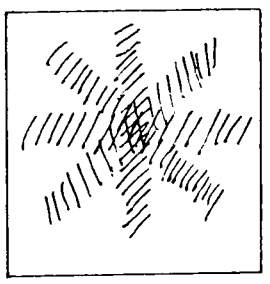


Fig 9.21
An input pattern that can cause an all 0's or all 1's
N tuple state to be generated

respectively. If one N tuple in the N tuple output image shows a `|` on top of `\`, this means that edge operators 0 and 135 degrees lie in the upper rank (and by default the 45 and 90 degree detectors lie in the lower rank).

Image op8 clearly shows the outline of the square in the tuple codes. The dark corners in the peripheral regions of the image are due to camera non linearities; they were not present in the scene. The N tuple patterns reflect this illumination gradient.

Pattern op1 is much more complex, and contains a lot of detail not resolvable by the edge operators in this low resolution view (see appendix 3.1 for a description of edge operator sizes). Careful inspection of the N tuple patterns will reveal the outline of the head.

9.7. Extending the N Tuple Process to give Confidences

The binary tuple process described above used 4 edge operators to detect 14 illumination gradients. The images shown in op2 illustrate a limitation of this process when applied to grey scale input images. Images Ai in op2 show a representation of the states generated when the binary tuple process is applied to a binary image of a square. Image Bi illustrates the result of the same process when applied to a grey scale image of the same square.

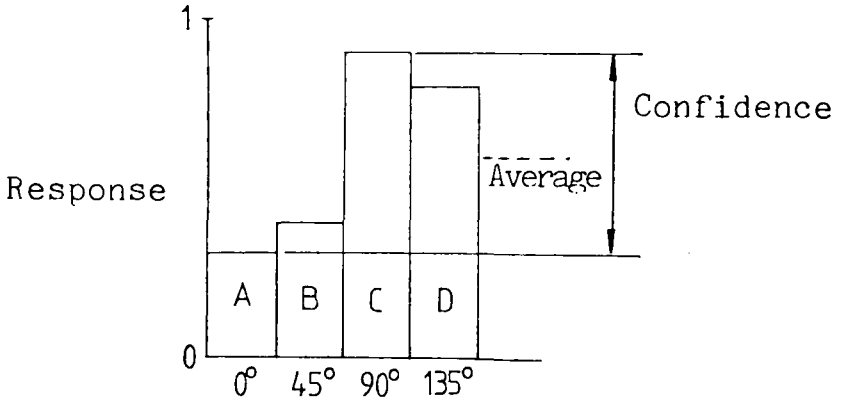
The regions in image Ai where no N tuple state is indicated illustrates areas where all the edge detector responses were equal (all 0's or all 1's). As explained in the previous section these states are not given N tuple states. In image Bi, which is processing a grey scale image, all edge detector groups are assigned a state (i.e no all 1's or all 0's states are present). This results because there are no

uniform contrast regions in the grey scale image thus no edge detectors respond with a zero illumination gradient (required to produce all 0's N tuple states).

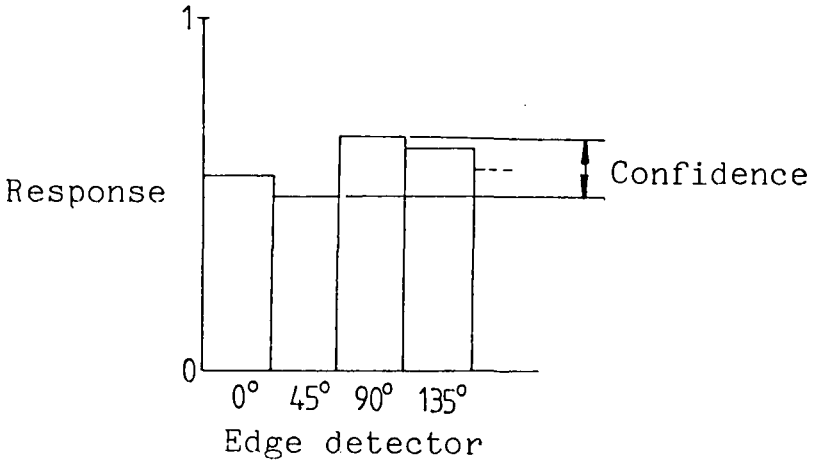
The following can be deduced from images Ai and Bi in op2. An N tuple state is an all or nothing response to a contrast gradient viewed by the four tuple of edge detectors. Even if the illumination profile viewed by the N tuple sample of edge detectors is only slightly uneven an N tuple state will be generated. In binary images this does not cause any problems because large areas of the image will have a uniform illumination gradient. However, in grey scale images this is a problem. Edges of shapes as well as clear areas of the scene will generate particular N tuple states, effectively indicating edges present in areas of uniform illumination. This would adversely effect recognition in that the system would recognise shapes present in regions where no shapes are actually present.

To overcome this problem each N tuple state is given a confidence value, calculated from the `quality` of the pattern the edge detectors are `seeing`. This is passed to the memory to aid recognition. (How this is used by the memory is shown below). A group of edge detectors viewing an edge may give rise to an N tuple state of 1010, with an associated confidence of 0.95 out of 1.0, whilst an area of near uniform illumination might generate the same state but with a confidence of 0.1.

Fig 9.22 shows how the confidences may be calculated. These graphs are representations of the edge detector responses for each of four tuple group of edge operators. To calculate the N tuple code the average response of the maximum responding and minimum responding edge detectors is used and then ranking occurs (see above). The calculation of



	Top	Bottom	Confidence
N tuple	CD	AB	0.5



	Top	Bottom	Confidence
N tupe	CD	AB	0.15

Fig 9.22

confidences is simple, and is given by ; Average response of edge operators ranked top - Average of edge operator responses of edge operators in the bottom rank.

This gives a confidence value independent of overall response of the edge detectors and so gives a true reading of the presence of a particular N tuple state. The result of this process is shown in op2, Aii and Bii for the same images as before.

The process above has been termed confidence tupling, the process used to find the N tuple codes is called binary tupling. The use of the confidence output by the memory will be described later.

9.7.1. Recognition of a `star` or Uniform Intensity Profile

It was shown earlier that a problem existed in assigning ranking to a set of equally responding edge operators. The binary tuple process gave the same ranking to a ´star´ pattern and a uniform grey level pattern. It was required that the ´star´ pattern be represented in the N tuple states but not the area of equal grey level. By using confidences this requirement can be met. As a uniform grey level area would give all edge operators a zero response, thus the confidence for this pattern in all N tuple samples would be zero. Conversely the star pattern would produce non zero responses from the edge detectors, thus a non zero confidence for some N tuples. For all 1`s n tuples and all 0`s n tuple the average responce of each edge detector in the n tuple is used for the confidance of the n tuple. By correlating the confidences of a N tuple with the N tuple state the memory can be made to ignore all uniformly illuminated areas.

For interest op 4 has been included. This shows the confidence tuple process acting on the same image as used in fig opl. To show up the outline of the face the confidences of the N tuple samples have been thresholded at a suitable level. This has then been used to select N tuple samples for the N tuple output image also shown.

The effect of changes in illumination on N tuple confidences is shown op5. This is a view of the star image shown in op6. The top image differs by three camera `f` stops compared to the bottom. It can be seen that the N tuple patterns still show a good relation to the input pattern, although the N tuple confidences have fallen dramatically.

9.8. Using N Tuple Confidences in the Memory

The following formally describes how the confidences of each N tuple state are used by the memory described in chapter 6. If we consider the mathematical representation of the memory system given in chapter 7,

R_i = picture vector

C_j = class vector (teach)

CR_j = class response vector

where

$0 < i <= Rmaximum$

where Rmaximum = Size of the picture array R.

$$= 2^n \cdot \frac{P}{n} - 1$$

where P = no of pixels in input image

n = number of edge operators or n tuple size

and

$0 < j <= Cmaximum$

where Cmaximum = size of class array

and now

Q_i = vector of confidence values for N tuple states.

M_{ij} = Memory array

R_i, C_j, M_{ij} can take the values 0 or 1

Q_i can take the values 0 To 1

CR_j can take the values 0 to n before N point

thresholding and take 0 or 1 after

Teaching dose not use the confidence tuple information Q_i

thus it is the same as before

$$M_{ij} = R_i \cdot C_j \text{ for all i and j}$$

Testing or recall is now altered to

$$CR_j = \overset{i=Rmaximum}{\underset{i=1}{\cdot}} Q_i \times R_i \times M_{ij}$$

The teaching process requires the inputs to be 0 or 1 to allow the switch of a memory location to be determined. Alternatives incorporating the confidence values would be the subject of further investigation and could involve probabilistic switching of the memory elements depending on the confidence values of the input N tuple and possibly the use of a linear memory matrix instead of a binary matrix.

9.9. Summary

to summaries, the following attributes of the system have been discussed.

All 0's N tuple states are not implemented in the memory.

This overcomes the memory saturation problem and allows iterative recognition of objects, starting at the largest.

N tuple groups of edge detectors are distributed locally.

This allows logical relations between elements of line drawings to be recorded, and reduces interference in N tuple states in cluttered scenes.

Two fields of different resolution are implemented in the input window.

This allows global as well as local logical relations to be detected in the input scene.

A simple Roberts edge detector is used as an input to the N tuple process.

This is fast to compute and can be made sensitive to a preferred orientation.

A ranking process is used to assign a state to each N tuple group of responses.

This records the illumination landscape over the N tuple sample for grey scale images whilst being insensitive to the overall lighting level in the scene.

A confidence is assigned to each N tuple state.

This allows edge features to be detected in presidence to slight background illumination non-linearities.

The use of N tuple confidences in the associative memory has been defined.

The recognition of grey scale scenes perhaps containing many objects where lighting conditions are not consistent is a central problem in scene analysis. The aspects of the scene analysis system described above were designed to overcome these problems. Furthermore, the ability to recognise line drawn and block filled objects is a novel aspect of the system.

The methods by which the human visual system is able to cope with these problems is central to the studies of visual cognition (Pinker1985). The methods used above incorporate many features of the

human visual system (i.e foveation, input image resolution changes and edge processing) and may lead to a better understanding of the ways humans overcome the problems.

From now on the normal N tuple process will be called the `random mapped` input process, and the ranking system described above as `binary tuple` input process.

CHAPTER 10

Position Independent Recognition

10. Introduction

The discussion in chapter 3 pointed out that N tuple pattern recognition only recognises shapes at the position in the input window that they were taught. To overcome this limitation the image can be taught as it is moved around the input window. Unfortunately this leads to a larger teach set and broader generalisation.

This has been overcome by using a preprocessor to align the object before teaching and recognition of the object. Many methods of alignment are possible, however, alignment on one object in a scene containing many objects poses specific problems. A method derived from studies on the human visual system has been used. The human visual system is able to select objects present in the peripheral visual field for attention, and align the eye so that objects fall at the centre of the retina or fovea of the eye. The visual system appears to use a simple criteria to select which parts of the scene are to be foveated. Work done by Yarbus (Yarbus1967) and by Makworth and Morandi (Makworth1967) suggests that we attend to areas of 'high information' content. No exact definition of what information may be but it is suggested that it is where the illumination profile varies widely. A similar idea has been used in the system described here to provide position independent recognition.

It will be evident from chapter 9 that a process of foveation is needed to align areas of highest detail in the high resolution window. These problems are discussed in later chapters.

10.1. Selection of Areas of Interest

If we are able to detect illumination independent features, we can use these points to align the input window during teaching. During testing the same points can be used to recentre the shape. Moravec (Moravec1977) used an 'interest operator' similar to the one described below in an automatic obstacle avoidance system for a mobile vehicle. This operator detected any objects in the vehicles path, prior to recognition of the object.

In the selection of an area of interest there has to be a high probability of it still existing when lighting levels change and other objects enter the scene. An area of interest having such a property can be said to be highly salient. If an area of interest mingles into the scene when small lighting changes are made, the feature selected has low saliency. Furthermore, by assigning a value of saliency to each feature selected allows the system to recognise the objects quickly. i.e a system can be taught with highly salient features at the centre of the processing window, these features can be found during recognition and recentred upon. A low saliency feature may not be present when lighting levels change.

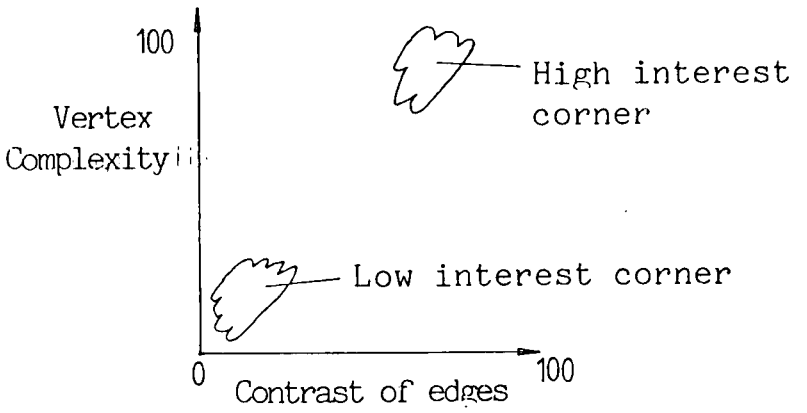
Thus in recognition the system windows on areas of highest saliency first and if recognition fails at these points moves to other features of lower saliency until recognition succeeds.

10.2. Detection of Highly Salient Features

A highly salient feature of scenes is a contrast gradient. High contrast gradients within a scene are less sensitive to illumination changes than low contrast gradients, in that a low contrast gradient becomes undetectable if illumination of the scene is reduced.

Quantification of contrast gradients are already made by the edge detectors which are used to reduce the image to an edge description prior to N tuple sampling, thus they can be used to detect areas of interest. Vertices are also used to quantify saliency in the system. Vertices with many edges converging are given a high saliency, while simple vertices are given a low saliency. Other similar features which could be used are colour gradients and changes, and motion gradients. These latter features have not been explored here.

The two measures, contrast gradient and vertex complexity, can be expressed in an interest map as shown in Fig 10.1, which illustrates how the two measures may be combined. This illustrates a shape with various measures of interest, the `interest` map is shown at the top with the shape represented below it. The top right hand corner of the shape shown has a vertex with two edges meeting and with high contrast for both its edges, thus it has a high saliency as shown in the



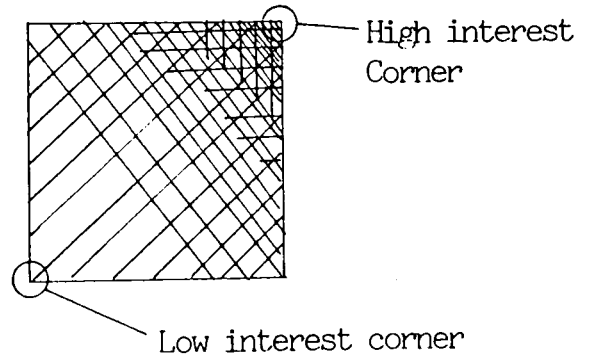


Fig 10.1
Interest area maps

interest map. The bottom left hand corner of the shape has the same vertex complexity but lower contrast and thus has a lower saliency. This is depicted in the interest map at the bottom right hand corner. Other parts of the shape may be mapped in a similar way.

10.3. Design of a Saliency Operator

A saliency operator that did not forefil requirements is shown in Fig 10.2, this is made up of 4 edge detectors at different angles. Each edge detector quantifies the contrast gradient in one dimension (a grey scale input image is assumed). The formula used to compute the contrast gradient by each edge detector is,

$$R = |r(a) - r(b)| + |r(b) - r(a)|$$

The values for r(x) are given by the sum of the responses of individual pixels in the regions shown in Fig 10.3, which illustrates one edge detector.

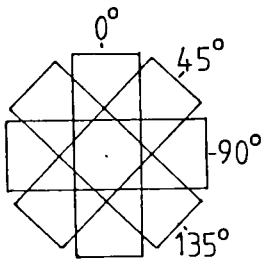


Fig 10.2
A saliency operator

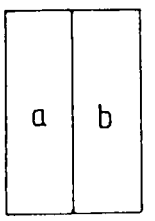


Fig 10.3
Edge detector construction

The simplest function to compute the saliency measure 'I' from the group of four edge detectors is :

$$I = R(0^{o}) + R(45^{o}) + R(90^{o}) + R(135^{o})$$

where R(X) = Response of an edge detector at an angle X.

Unfortunately this function does not give a good measure of vertex complexity although it deals with the contrast gradient well. Fig 10.4 shows the responses of a group of detectors viewing a simple corner vertex and a simple 'edge', and gives the saliency value for each. The saliency values (I) are almost identical in both cases (1.6 compared to 1.3), although the vertex complexity differs. It was considered that the difference between these values was not great enough to allow accurate repeatable foveation to take place. It was important to have a clear difference in I for an edge and a simple vertex. If this were not so the system would need to foveate on all edge pixels to provide fast recognition.

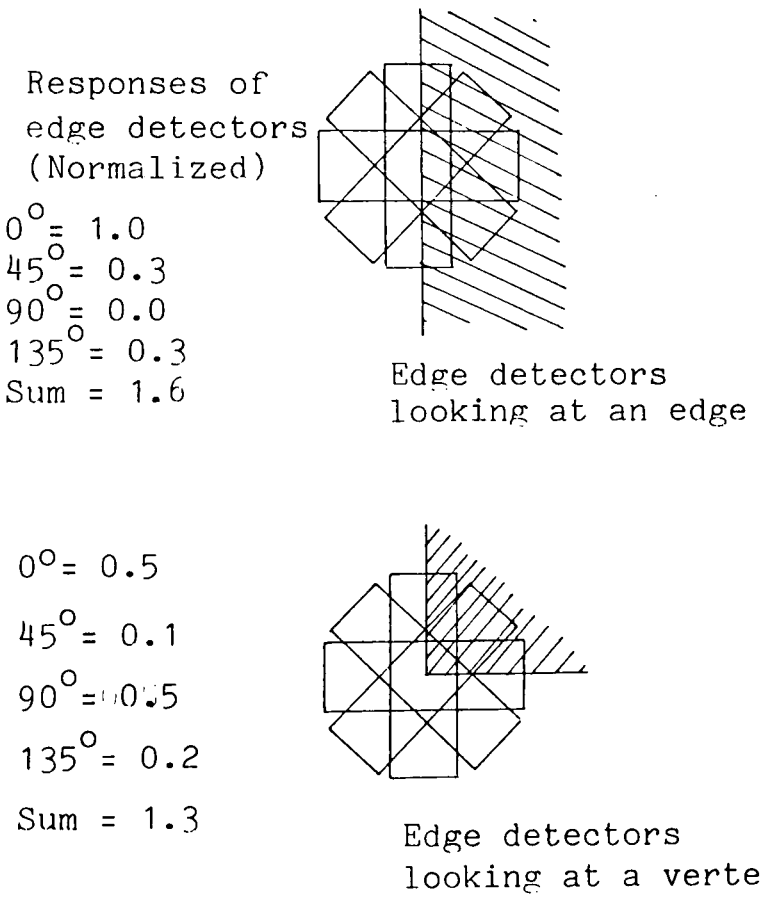


Fig 10.4
Response of the saliency operator for
two different edge constructs

Fig op9 illustrates the use of this operator when applied to the image in op7. The responses are shown as raw data and as a grey scale image. It can be seen that the vertex of the square has a lower saliency value than the edges as predicted (see Fig 10.4).

To overcome this an alternative process was used which used the primitive processes that make up the edge operators used in the N tuple sampling process (i.e to maintain speed). The new saliency operator was specifically designed to give a maximum response to a simple vertex. This is shown in Fig 10.5

A vertex is only present when both the x and y axis shown in Fig 10.5 cross an edge. To detect an edge passing through either axis the following functions are used. These operate on the sum of the response of pixels in the regions a,b and c shown in Fig 10.5.

$$\text{detector } x = |b - a| + |a - b|$$

$$\text{detector } y = |a - c| + |c - a|$$

The responses for x and y are multiplied to give a single measure of

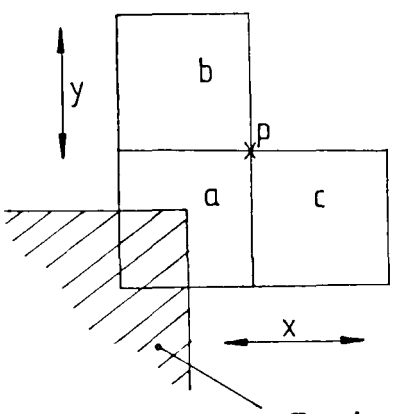


Fig 10.5
A simple saliency operator

vertex strength. To detect a vertex over a range of angles the same function is applied to a detector rotated in 8 equal increments around P (in Fig 10.5). The total information measure value is given by averaging all 8 response from all orientations.

The result of this process is given in Fig op10 on a 128 x 128 pixel grey scale image given previously. The top shows a square with responses for the corners greater than the surrounding edges. Over the whole image the bottom left hand corner gives the highest response.

The other two images are for the star pattern in op6 and the face in op1. In the star image a high response is given for the centre of the star which is what is required. In the image of a face the outline of the head has been selected as the most interesting, with the highest responses for the top of the hair line, internal features give a lower response.

Fig op10 shows that the detectors are sensitive to changes in lighting levels. In this image the top right hand corner is less well lit than the bottom left hand corner, as a result the latter gives a higher saliency value than the former. In some circumstances this may cause the system to concentrate windowing at the brightest area of the scene. However, because saliency is measured with respect to the contrast values of a set of edges this result is unavoidable.

This operator is also sensitive to small off-sets in the image. Fig op11 illustrates this, the image being viewed is the same as that in op10 top, but the image has been off-set by -2,-2 , +2,+2 and -2,+2 pixels respectively. It can be seen that the information measure response varies widely. This can be explained by the movement of edges within the detector group. However apart from the difference in vertex responses due to lighting , the responses to vertices are

constantly above surrounding values.

To combat any missed vertices due to lack of resolution four samples are made at n pixel offsets, the responses from each edge operator are then averaged, where n = detector element size / 2.

10.4. Summary

A process for measuring a particular position and lighting invariant feature within an image has been defined. This allows translation independent recognition to be performed by shifting the centre of the processing window to the feature. It has the property that simple vertices have a higher information measure than straight edges, and higher contrast vertices can give a similar but higher measure. The process has been designed to incorporate processes already being performed in the calculation of edge operators thus it effects the processing speed as little as possible.

CHAPTER 11

Image Acquisition and Recognition Strategies using a Movable Window

11. Introduction

This chapter considers the image acquisition and recognition strategies used to recognise complex scenes using a movable window. It also considers the means by which high and low resolution windows described in chapter 6 are connected to the associative memories.

The format of the system described so far is shown in Fig 11.1. The scene is imaged by the camera to form a grey level image array on which N tuple processing is performed using one of the methods described in chapter 9. This results in a key pattern which is used to access and teach the memory section. The memory section is formed from two associative memories as described in chapter 6, the first associating from the picture to the class such that in recognition a class pattern is generated. A 'class' pattern is an individual label or identifier which identifies each particular input pattern. The

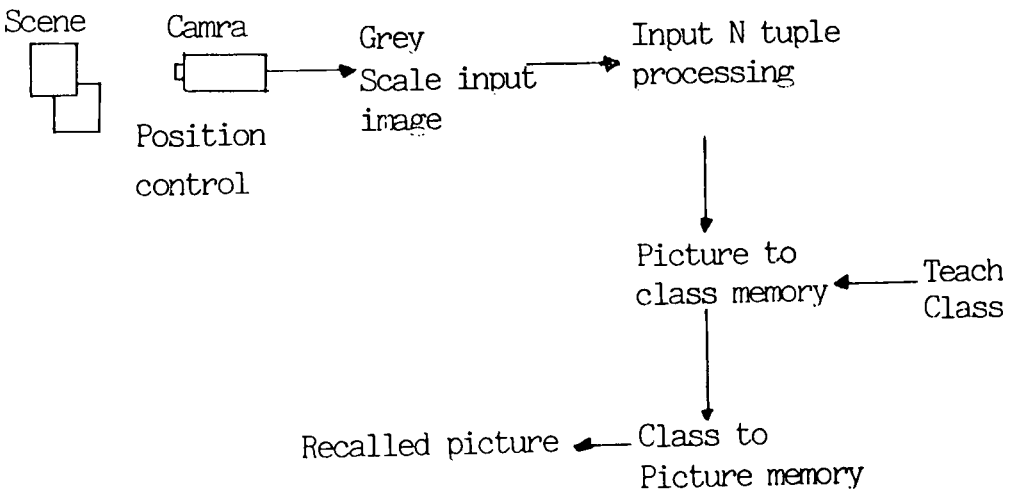


Fig 11.1
Format of the recognition system

second memory associates from the class pattern back to the picture such that a complete image of the object in view can be generated for occlusion analysis.

11.1. Connecting the High and Low Resolution Windows to the Associative Memory

Chapter 9 (input processing) introduced the use of a high and low resolution window to allow global and local relations between features within shapes to be detected. The problem of coupling two windows to the associative memory was considered with the following conclusions. There were two possible ways of implementing this using either one or two associative memories. If one is used the two input fields generated by the separate high and low resolution windows are combined into a single input array, using one class pattern for both low and high resolution images. If two associative memories are used, each input window is served by one memory, and each window has its own class pattern. Since the latter construction could be easily reduced to the former by providing the same class pattern to both memories and combining the class response outputs, two associative memories were used. This arrangement is shown in Fig 11.2. It can be seen that this results in two parallel systems. This can be extended to more windows by the addition of another system, as might be the case if more resolution is needed.

11.2. Teaching and Testing Strategies

We now consider the teaching and recognition of images. In this section the discussion concentrates on a system using the low resolution window only. Integration of data from the high resolution window is

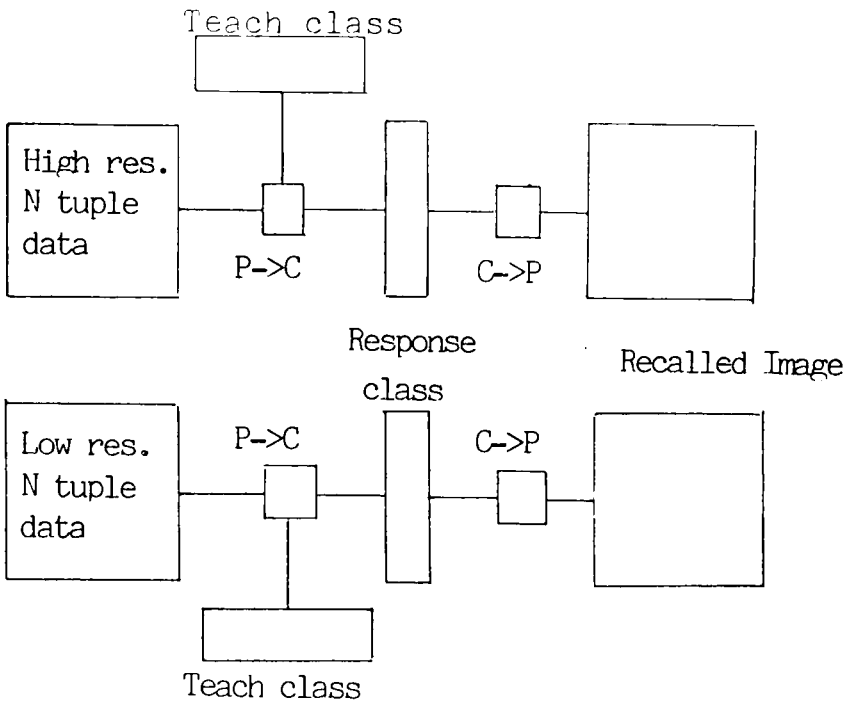


Fig 11.2
Using parallel systems for each resolution of window.

considered later.

Although it will become apparent that the process of teaching and recognition by the system must interact, the two process will be considered separately.

11.2.1. Teaching

The process of teaching involves the acquisition of the shape into the memory such that it can be successfully recognised and recalled later. Only case where the object to be learned is present within the image clear of any other objects is considered.

Consider a simple teaching strategy

1) Find positions of high interest.

2) Move the window to centre on each area of high interest.

3) Teach the image at each position of high interest with a new randomly generated class.

4) Stop when all high interest regions have been visited.

This process will teach the object at recognisable high interest areas so that in recognition the object can be successfully recalled.

Problems arise with such a simple procedure. First there is a problem if shapes only differ by a small amount. This may result in two patterns only differing by one N tuple code. As a result there is a high probability that during recall these two patterns will be confused. This applies to learning two simple patterns as well as teaching the same pattern over a number of interest points.

This is shown in the example in Fig 11.3. Patterns A and B have been taught at the position shown under different classes. The image to recognise has a small portion missing, which is the only part of the shapes which separate the two patterns into the two classes. Thus on recall the class pattern is confused, shown by the occurrence of both class patterns in the response class. The small part missing could be a small distortion between the two patterns, which would produce the same effect.

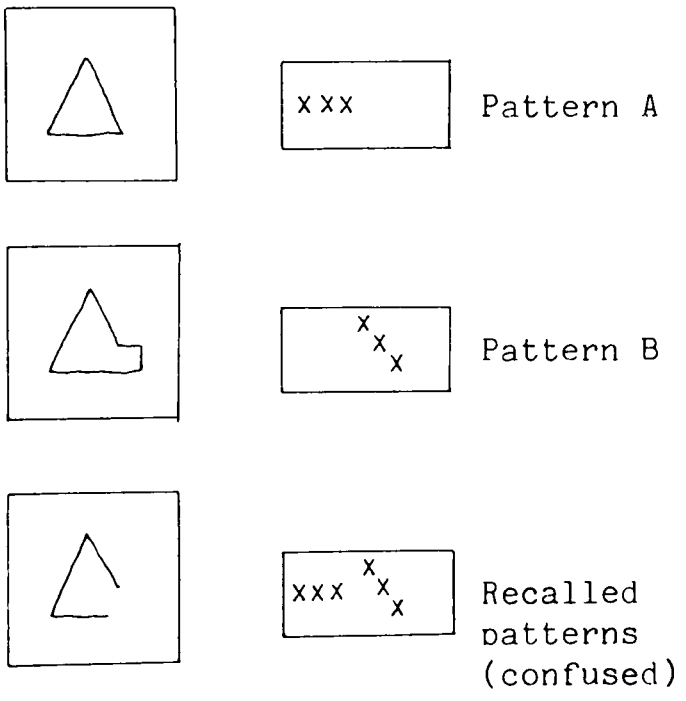


Fig 11.3
Teaching and recognising similar patterns.

At the extremes the patterns may only differ by a unit hamming distance. In this case only a small amount of noise would cause confusion, i.e an inability to get an N point class.

The problem can be overcome by teaching a new class pattern when the input and recalled pictures have a hamming distance greater than x. If the patterns differ by less than x the high resolution window is used to discriminate the patterns. The size of x would define the noise immune of the system. If x were large all patterns would be defined as the same class in low resolution. If x were very small then the problem sited above would occur.

In order to do the above it is necessary to find the closest pattern to the one presented then find the difference between this pattern and the input pattern. To enable this a test is done prior to a teach. If the difference found is less than x then the recovered class pattern (if correct, with N points) is used in teaching along with the input pattern, otherwise a new class is generated.

The process of using the recovered class as the teach class provides a generalisation process. Since many similar patterns will be taught under the same class a generalisation will take place between these patterns such that a larger set of input patterns than that taught could be recognised.

11.2.2. Teaching Strategy

The following teach process incorporates what was described above. (this is also shown in flow chart 1 in the appendix).

1) Find the areas of interest.

2) Saccade to a 'high interest area'

3) Test the memory at this point P->C then C->P.

4) Calculate the difference between the memory input and response from the memory.

5) If the difference found is greater than x then

5i) Create a new class and teach P->C and C->P memories.

If the difference found is less than x then

5ii) Using the recovered class teach the P->C memory only.

7) Repeat until all areas of interest have been visited.

When teaching on the recovered class at stage 5ii) there is a choice whether to teach just the `picture to class` memory or both this and the `class to picture` memory. If the class to picture memory is taught in this instance the result will be a blurring of the response picture when testing occurs. The result would be to make separation (i.e. difference greater than x) possible on images which do not differ by an amount greater than x. Thus to prevent this only the picture to class memory is taught. As a result generalisation occurs in recognition but not in recall.

11.2.3. A Strategy for Limiting the Number of Window Positions Taught

In the example above teaching stops when all high interest areas have been visited. In practice this would be to thorough, resulting in many views of the same pattern being taught. To reduce this number a threshold can be set on the number of points that should be taught at. An alternative approach is to monitor the level of activity (The complexity and contrast of the vertex) in the interesting area detectors and

only teach at a percentage of the highest responding detectors. Ideally there needs to be a process of verification which checks that the system can reliably recognise the object when slightly shifted after each P patterns have been taught.

11.3. Recognition Strategy

This process involves recognition of all shapes within the scene and recall of the associated patterns of each shape. Recognition is to be performed on complex scenes of many overlapping shapes. The basic process of recognising one object involves the following important aspects,

1) Recall of the associated pattern from the image presently viewed.

2) A level of confidence assigned to the recognition process such that it is known when recognition has failed.

3) An ability to place the recognised object within a class of shapes, i.e. find its identity.

Recall of the associated pattern is done in two stages as previously described. First the P->C memory is tested to recover the class, subsequently the C->P memory is tested to recover the associated image. It is important that that the class pattern recovered is correct (i.e. one previously taught) otherwise recalling the associated picture from it will be unsuccessful. Verification of the validity of the class pattern can be made by checking that N class points have been recovered, where N is the number of points set to one in class patterns taught. If the class does not contain N points at l recognition has failed.

The process above indicates a definite recovery failure if the class

points do not equal N. However, if the class points do equal N there is still a possibility that recall has failed. It is either due to saturation of the memory or because there is an unknown pattern on the input that is slightly like two or more patterns previously taught. (This is expanded on in chapter 12). To overcome this a process of picture confirmation is used. To do this the recovered picture is fed back on to the input array and re-tested to get another class image. If this class image is the same as the one previously recovered (on the original pattern) then recognition can be said to be successful; if the class patterns differ then the recognition can be said to have failed.

The whole problem of 'when can a system know if it has recalled a correct pattern' is very important. In the case above a process of confimation is used. This gives an all or nothing measure of correct recovery. Ideally a linear measure needs to be assigned, in the form of a confidence value which indicates how sure the system is in the recognition it has performed. This can be obtained by considering the response profile of the class pattern shown in Fig 11.4. The confidence is the difference in response between the minimum responding point set to 1 after N point thresholding and the highest other

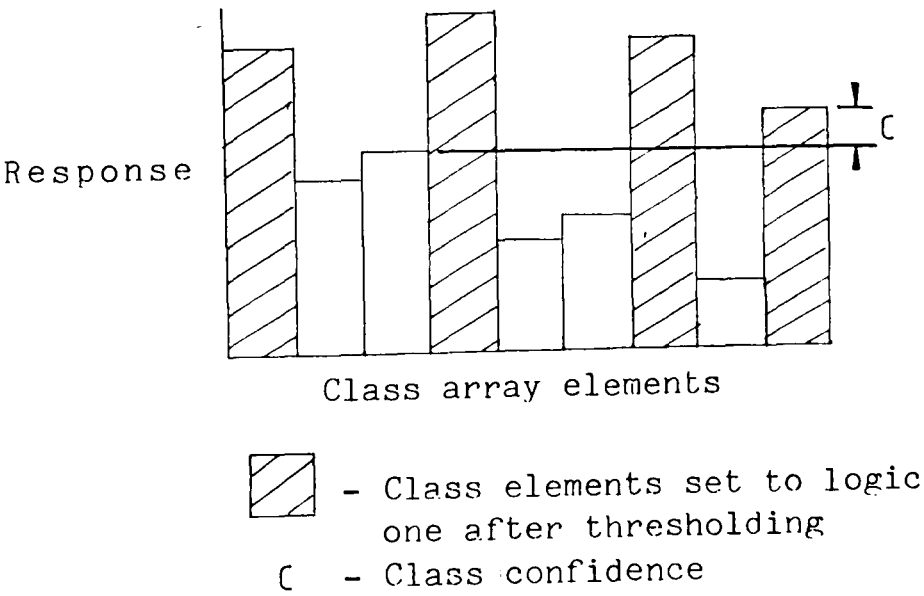


Fig 11.5
Deriving a `confidence` figure from the class pattern.

element responding which has not been set to one by thresholding.

The value produced by this process is used to decide whether the class pattern is accepted or rejected. Unfortunately this requires the selection of a suitable threshold. The previous re-test process requires no threshold and is more sensitive and easier to use. The only problem with the confirmation method is that it requires a second test of the system which could be prohibitively slow, it is also shown in chapter 12 that this process is unreliable.

The test process can now be defined, see flow chart 2.

11.4. Providing a Label to Identify the Class of the Input Object

In the process above there is no facility for determining which pattern class the shape which has been recognised belongs to. Each view of every shape has a different class pattern assigned to it. This means that each pattern class has a set of class patterns that can identify it. To be able to assign a specific label to a shape, the set of class patterns belonging to the shape must be mapped to a specific label.

There are many ways in which this can be done, the class pattern can converted to a number and this looked up in a table which records which numbers relate to which classes of patterns. This process would be simple, although the conversion of the class pattern to to a number is problematic. This approach has not been adopted on the grounds of speed and uniformity. Since the output is a pattern it is quite straight forward to create another associative memory that links the class patterns under a specific shape to a single image or 'identifier image' that identities the pattern class to which the shape belongs. The use of this associative memory would provide a uniform system

where all units are associative memories instead of separate associative memories and look-up tables.

The only problem with this approach is the amount of storage required and the recall abilities when associating a class with an identifier pattern. In the other associative memories one pattern is N tuple processed and the other supplied directly to the memory. This produces an efficient system since N tuple processing reduces the ratio of bits at 1 and bits at 0 in the input to the associative memory. This means that only a small proportion of bits are set to one in the memory matrix compared to the overall size of the memory matrix. If the class pattern is associated with the identifier pattern it would necessitate the need for N tuple processing on either the class pattern or/and the identifier pattern to provide the same storage properties as is found in the other memories. N tuple processing need not be done if the response picture from the memory is used to associate the identifier in the identifier memory. This provides the same conditions as is found in the picture to class memory as long as the number of bits set to one in the identifier pattern is restricted to a particular value (to prevent instant saturation on teaching). Since the recalled picture will be exactly the same as the picture that was taught, the thresholding of the identifier pattern after testing can be a simple threshold of the highest responding points in the response array.

Fig 11.5 shows how the identity image is linked into the rest of the system. This shows a system where only one resolution window is implemented. When two or more windows are implemented the identifier pattern is the same for both systems (two identifier memories being used).

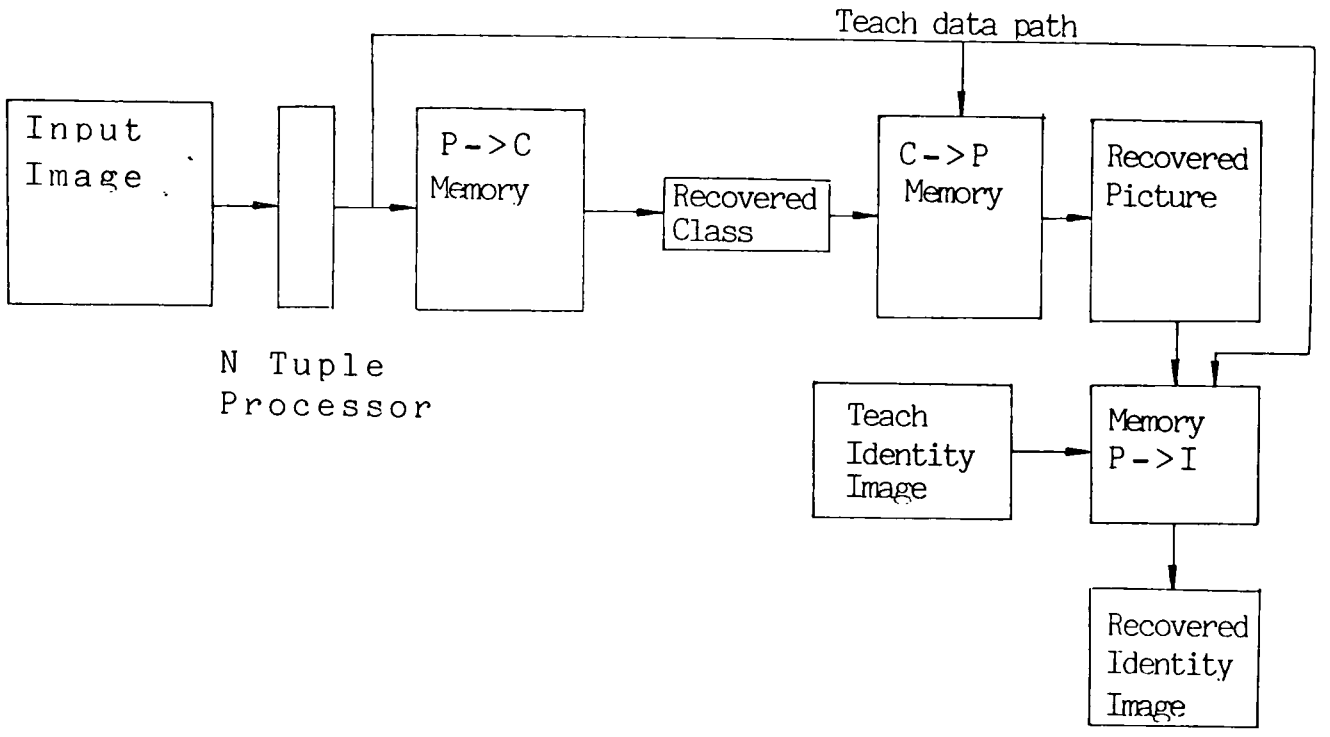


Fig 11.5
Single resolution system - integrating the identity image.

11.5. Incorporating the Identifier into the Teaching Procedure

Little alteration is needed to the present teach procedure to incorporate the identity image. When teaching a new shape the identity image is presented which belongs to the shape being taught. The input image is provided as the associating image for the identity pattern when testing occurs the input to the identity image is then switched to the output of the C->P memory.

11.6. Incorporating the Identifier into the Recognition Procedure

When successful recall has taken place as defined earlier the output image is supplied to the identity memory to recover the associated identifier.

11.7. Summary

The definition of a simple teach and test procedure has been given such that shapes can be learned and recognised using the low resolution window (see flow charts 1 and 2). We shall now go on to look at

how to determine occlusion and how the high resolution window can be incorporated to resolve the identity of the shapes which are confused at low resolution.

CHAPTER 12

Investigations into the Recognition Systems Performance using Natural Images

12. Introduction

This chapter gives the results of investigations performed on the recognition system using 'natural' images. The qualitative effects on recognition performance using different input processing methods during teaching and testing are reported. Recognition abilities are examined using only the low resolution window.

12.1. Description of the Recognition System

The investigations were performed using a grey scale image which is convolved with edge operators. This is then N tuple processed to produce the N tuple state data and confidences relating to these states. This data is presented to the first memory denoted `picture to class` or P->C. The output generated by this memory during testing is N point thresholded and then fed to the second memory, the `class to picture` memory or C->P. The recovered picture is sent to the identifier memory to recover the identity image for the object recognised. The constant system parameters which would allow these investigations to be repeated are given in appendix 4.

12.2. Description of the Investigations Performed

The results of the following three investigations are reported.

1) The first investigation explored the effects on recognition per-

formance for a system taught on N tuple data which was not thresholded and a system taught on N tuple data which was thresholded. Thresholding the N tuple data takes the following form. If a particular N tuple state is set and has a confidence that is above threshold it remains, if it has a confidence below threshold it is removed. The threshold is calculated as half the average confidence of all N tuple confidences of the pattern being windowed. More details are reported below.

2) The second investigation extended the results of the first investigation by examining the effects of using or not using the N tuple confidence data during recognition, as well as exploring the combined effects of using thresholding of the N tuple state data along with confidences during recognition.

3) The last investigation explored the effects on recognition success as more patterns are taught into the system.

12.2.1. Description of Image Data Used in the Investigations

Grey scale input images are used, consisting of grey scale shapes cut out from suitable card and placed on an uneven black reflective polythene background. Lighting is provided to produce a suitable depth of illumination over the grey scale to allow shapes to be well defined.

Test images used are shown in op12 as well as the originals in op13. Those in op12 contain two shapes, both show the triangle occluding the square. In the top image in op12 the triangle is darker than the square, in the bottom image the square is darker than the triangle. By using shapes with different grey levels the system can be `forced`

to centre the darker shape first. This is because of the nature of the ´interesting area´ finding process, various parts of the image are found more interesting than others, in the following manner.

Most interesting ..... High contrast vertices

High contrast edges

Low contrast vertices

Least interesting .... Low contrast edges

Only enough interesting areas were windowed on to maintain foveation within the shape required.

12.3. Positions of the Input Window at which Each Shape was Taught

In all investigations the images in op13 were used to teach the system. The `interesting area` process described in chapter 10 identified the points shown in op13 (crosses) for windowing at. The center of the processing window was placed at these points where upon the system was taught.

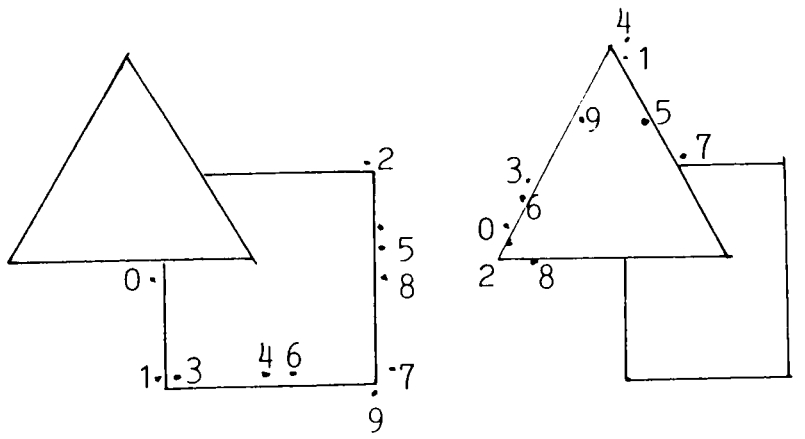


Fig 12.1
Positions and the order in
which the window was tested.

12.4. Parameters Noted in the Investigations for Each Test

At each window position tested, various parameters were noted concerning the response given by the system. The results of the investigations are given in tables 1 - 4 in the appendix.

The following typical data was recorded in investigation 1 after each test cycle (the label used in the results is given in brackets). The explanation of the entries in tables 2 and 3 are given along with the tables in the appendix.

Recovery of known identity image. (Identity)

Whether or not a known identity image was recovered, assessed by visual inspection.

Image out. (Image out)

Whether a recognisable shape was recalled on the particular test, assessed by inspection.

Image out analysis. (Response in : number of points at logical 1 in response and in input images / number of points at logical 1 in response that are not in the input image)

Two variables were recorded, the input pattern and the response pattern were compared. The number of points both in the input and the response image were noted along with the number of points in the response that were not in the input image. If recall was successful, this indicates how much of the input image was involved in recalling the image. The sum of these values gives the total number of states at 1 in the N tuple data array presented to the memory.

Class response analysis. (Worst case)

This is the confidence the system gives to the class that is recalled. See chapter 11 and Fig 11.4 for a discussion of this value. It is the difference between the maximum responding class point set to 0 and the minimum response of the class point set to 1 after N point thresholding.

Class response analysis. (Class analysis)

The left value of each entry is the average response of class points that remain at one after N point thresholding. The right value is the average response of class points that are switched to zero after N point thresholding.

Class points recovered. (cl. p. 1 and cl. p. 2)

The number of class points at one recovered on the first test (cl p. 1) and second test (cl p. 2).

12.5. Investigation 1 : Into the effects of teaching non thresholded and thresholded N tuple data

This investigation compared the recall ability of the system after teaching the system on thresholded or non thresholded picture N tuples. N tuple confidence data was supplied to the memory in all test runs (as described at the end of chapter 9).

12.5.1. Teaching Procedure

The images shown in op13 were presented to the system along with the identifier patterns shown in Fig 12.2. Each pattern (one of a square and one of a triangle) was taught at 10 window positions, with a different 4 bit random class generated at each view. The teaching procedure is shown in flow chart 1.

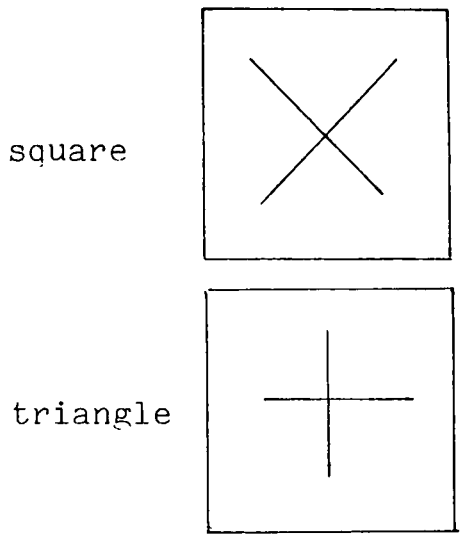


Fig 12.2
Images and Identifiers used in investigation 1.

Two different teach process were performed, one in which the N tuple confidences were thresholded at the average confidance level of all n tuples and one in no thresholding occurred.

12.5.2. Recognition Procedure

Tests were performed after teaching the system on thresholded N tuple data and then after teaching the system on non thresholded N tuple data. In this investigation the square was presented (opl3 top) to the system and tested in the same 10 window positions at which it was taught. The input N tuple pattern was thresholded in all cases, and N tuple confidence data was supplied to the memory.

12.5.3. Findings of Investigation 1

This investigation illustrated the effect of teaching the system non thresholded and thresholded data. Two patterns were taught to the system a square and a triangle. Testing was only done on the square. The results of the investigation are shown in table 1.

12.5.3.1. Window Positions at which the System Taught the Image

As explained in chapter 11, during teaching a pre test is done to see whether an acceptable number of N tuple states differ between the response image at that position and the present image, if an acceptable difference is found the present image is taught. In all investigations the acceptable difference was set to 20. All teach views obtained a difference greater than this, and as a result all window positions were taught at.

12.5.3.2. Effects on Class Recall

The first observation is that the class pattern is recalled correctly with 4 class points recovered after N point thresholding. This indicates that the `picture to class` memory recalls which pattern is associated with which class successfully. However, a 4 point class which is different to the original taught could be occurring (this is explained below). The only way this can be checked from the results given is by inspection of the picture and identity image recovered. Recall of these patterns is discussed below.

The results also show that the class is more `confidantly` recovered when the N tuple data is thresholded prior to teaching i.e the `worst case` values in table 1 give an average response of 44.9 for `non thresholded` teach data and an average response of 52.9 for `thresholded` teach data indicating it is better to use thresholded N tuple data in teaching. This is due to less saturation occurring in the P->C memory when the input image is thresholded (less bits set in the memory on every teach).

12.5.3.3. Effects on Image Recall

The image output could not be visually recognised in the case where the system is taught on non thresholded data. This is because a shape can only be visually seen when thresholding occurs. For a system taught on non thresholded teach data the results for `points in the response not in the input` is high because testing is done on thresholded data (i.e the recalled image is non thresholded and the input image is thresholded- thus even if the recalled image was perfect a difference between the input and output image will occur).

The results for a system taught on thresholded data show slight differences between taught and tested images (small values occurring in the table under the heading ´points in response not in input´). This indicates that slight saturation is occurring in the C->P memory.

12.5.3.4. Effects on the Recall of the Identity Image

When the system is taught on thresholded data the identity image is recovered correctly on every test, when taught on non thresholded data the system recovers the identity image only six out of ten times. This indicates that saturation is occurring in the identity image memory when teaching a system on non thresholded N tuple data (since the response image was being recovered successfully). This is a great problem in using this type of teach data. Since the identity patterns contained a high proportion of bits set to one compared those set to zero and because identity patterns remain the same over many teach cycles this memory is likely to saturate quickly. To alleviate this thresholding the input image prior to teaching can be used, the results for a thresholded input image support this.

 (cont on P.176)

PAGE NUMBERING AS IN THE ORIGINAL THESIS

12.5.3.5. The Effects of Testing on Thresholded N Tuple Data

In all recognition tests the input image N tuple data was thresholded. This retained all N tuples relating to well defined features of the shape such as those for edges and vertices of the object, and removed N tuples responding to the background and solid body of the shape. The problem with thresholding during testing is that it may remove N tuples necessary for discriminating very similar shapes. This is not a problem here since the high resolution window can be used to detect these areas as will be explained later.

12.6. Investigation 2 : The effects of testing on non thresholded N tuple data combined with the use of confidences

This investigation explored the difference in recognition properties when using N tuple confidence data during recognition, as well as the

12.6.1. Teaching Procedure

The teaching for this investigation was exactly the same as for investigation one. Two patterns were taught, a square and a triangle, at 10 window positions each (images taught are shown in op13). The recognition investigations were applied to a system which had been taught on thresholded N tuple data and then to a system which had been taught on non thresholded N tuple data.

12.6.2. Recognition Procedure

Testing the memory was performed on the images shown in op12, visiting the points shown in the images in the order shown in Fig 12.1. Ten

window positions were tested at for each shape. The following variations in teaching were looked at,

1) Test images

Light triangle on dark square (Figure op12 bottom)

Dark triangle on light square (Figure op12 top)

As explained earlier, this arrangement means that the system effectively confines its windowing in the vicinity of one shape. In the first case the window is confined to the triangle, in the second case the window is confined to the square.

2) Tuple confidences used in testing or not used in testing.

3) Input tuples thresholded or not thresholded.

All combinations od the above parameters were investigated as shown in the table below. This gave 8 different test runs. Furthermore, all 8 tests were performed on a system taught on thresholded N tuple data and then on a system taught on non thresholded N tuple data. The results are given in table 2 and table 3 respectively in the appendix - note that some results from table 2 are repeated in table 3 for comparison.

Test	Threshold	Mem type	Image
A	NO	Con	Wt on Rs
B	NO	Bin	Wt on Rs
C	NO	Con	Rt on Ws
D	NO	Bin	Rt on Ws
E	YES	Con	Wt on Rs
F	YES	Bin	Wt on Rs
G	YES	Con	Rt on Ws
H	YES	Bin	Rt on Ws

Test : Test number.
threshold : Whether or not thresholding
was applied to the input N tuple data.
Mem type : Con => Confidences of each
tuple were supplied to the memory.
Bin => confidence non
supplied to the memory.
Image : Pattern type presented
Wt => light triangle
Ws => light square
Rt => dark triangle
Rs => dark square

Wt on Rt, A light triangle appears on a dark square - effectively forcing the system to confine window positions to the vicinity of the triangle.

12.6.3. Expected Results for Investigation 2

By considering the window position during teaching and its position during testing a prediction can be made as to whether the triangle or the square should be recognised. By doing this it is possible to say whether the recognition results are far below or above what might be expected. In the table below OK is placed in a row if the response would be expected, since the point visited by the window during testing is close to one visited during teaching. A `?` indicates that it is not clear which object should be recognised.

Window position	Pattern Description Dark square Light triangle (Respond as Triangle)	 light square Dark triangle (Respond as square)
0	OK	?
1	OK	OK
2	OK	OK
3	?	OK
4	OK	?
5	OK	?
6	?	?
7	OK	OK
8	OK	?
9	?	OK
	Total 7	Total 5

From the table it is apparent that the triangle will be better seen than the square. (as explained earlier).

12.6.4. Results of Investigation 2: For thresholded teach data

The results in table 2 are for tests performed on a system taught on thresholded N tuple data. The set up for each investigation was given above.

Although this investigation was small some conclusions can be gained. The first is that the triangle is recognised correctly more often than the square, as predicted above. Correct recall can be seen in the correct recovery of the relevant identifier and image (identifiers, square = x and triangle = +).

The results also indicate that the use of confidences in recognition results in both more successful recognition and higher class confidences. Tests A,E,C and G all used N tuple confidances during testing, the results show appreciably higher success in recalling the image than in tests B,F,D and H where N tuple confidences were not used. In all these cases the worst case class distance was greater when confi-

dence data was supplied to the memory on each test.

The results for the `number of points in the response image also in the input` and `the number of points in the response not in the input` gives an indication of the peripheral noise in the input image during testing. Given that the input image has been correctly recovered, the results for run A give the average number of N tuples in the response that were not in the input image as 26 and the number of N tuples belonging to the shape as 15. If these two values are summed the result is the total number of active N tuples in the N tupled input picture (after thresholding and out of a total possible of 64). The results show a high degree of noise in all images due to occluding or occluded shapes. When the recall of the image was not successful the response image was unrecognisable. This is reflected in the low responses for image out in the results.

The effects of thresholding the input image on the recognition ability of the system can now be looked at. Consider the tests where confidences were used i.e in tests A,E,C and G. Little appreciable differences in either successful recognition or in class confidences are found when thresholding is or is not performed during recognition. The results for runs A and E, E show more success at obtaining a clear picture of a triangle, but since this test is objective, i.e by visual inspection, it is not a reliable measure. Considering the average worst case class response between A and E, A shows a larger difference than E, but if we consider test C and G the opposite is true, G shows a greater difference.

Thus the first set of results show that using confidences in recognition does seem to improve recall success, while the effect of thresholding the input pattern during testing is inconclusive.

12.6.4.1. Failure of the `re-test` procedure for verification of recognition success

In chapter 11 a process was given which verified whether correct class recovery had occurred. To do this a process of retesting the response image was used. If the class resulting from this test and the original test was the same then class was successful. Inspection of the results found this process not to be successful, the class patterns remained the same in both tests even though recovery of the complete image was unsuccessful.

The following is one reason why this process failed. Consider Fig 12.4. This shows a typical image of a triangle and square, along with two class patterns which have been taught in to a typical memory. The areas highlighted between the two images indicate the position where both images produce the same N tuple states. all other areas producing N tuples with different states.

A test image is presented to the system, i.e. a distorted triangle as shown, with the areas highlighted forming N tuples common to the areas highlighted on the taught images. Because the tested pattern `i` only contains N tuples in common with the test pattern at positions where N tuples were the same in both test patterns, testing on `a` will produce a class pattern which is the coalition of taught class patterns (as shown). Although this class would be rejected as having greater than two class points active after thresholding, it can be seen that a C -> P test on this class would result in a pattern containing the common parts of the input pattern and the two taught patterns. Furthermore a retest on this recovered image would result in the same class pattern as before, which illustrates how this retest verification process fails.

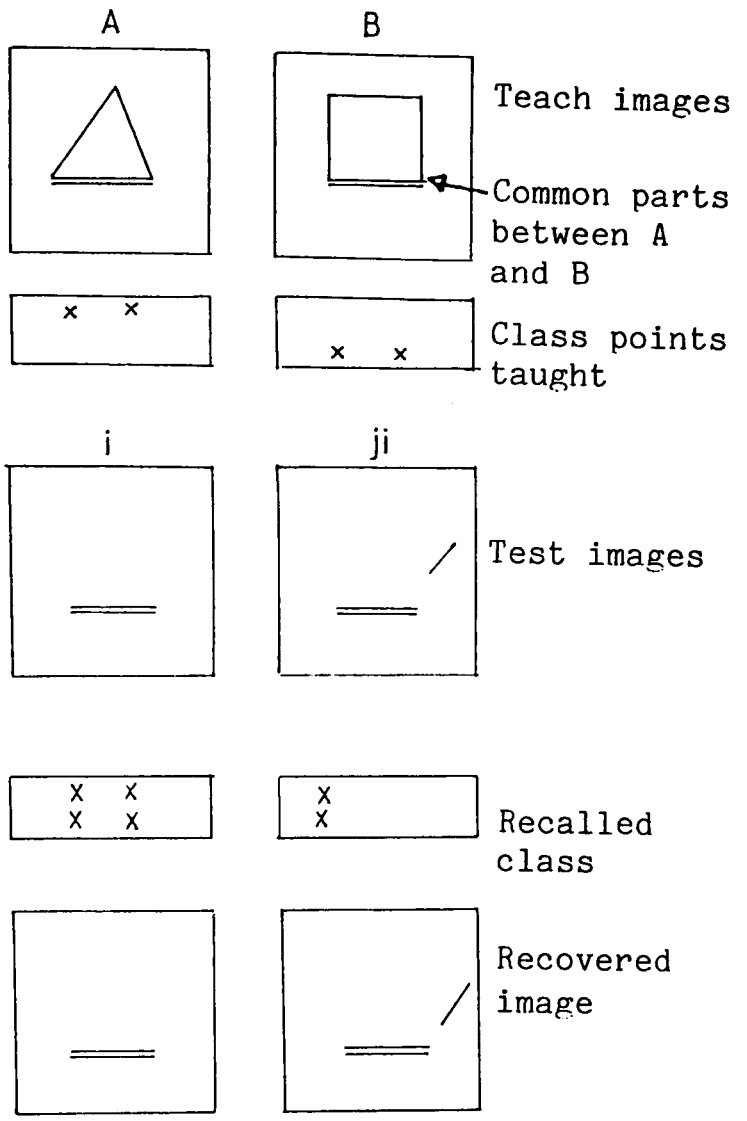


Fig 12.4
Figure to illustrate class point recovery.

In the above case above the recognition could be recognised as failing because a two point class image was not recovered. Unfortunately it is possible to get a correct number of class points in a class pattern which has not been taught, and still retain this same class pattern on a retest (as has been found in this investigation). This is illustrated in `ji` above . In this image the input is the same as in `i` but now there is a separate set of pixels active which generate N tuple states which are not present in the teach images A and B, but ARE present in other images taught into the memory. These states will have the effect of raising the response of two of the previous four points above threshold in the class pattern shown for `i`, thus now N point thresholding will result in a correct class pattern with two points active. This now means the recall cannot be rejected. The response image will again show the areas common to the two class

points in the response, and again a retest on this recoved image will not result in a different class pattern.

There is no simple way to alleviate the above problem, although the retest process works in some cases and checking for the correct number of class points also helps. The way this problem was overcome is covered in the next chapter as it involves the occlusion analysis process. It basically consists of averaging the response from the memory over a number of views to smooth out occasional inconsistencies.

12.6.5. Results of Investigation 2: For non thresholded teach data

The investigation above was repeated on a system that was taught on non thresholded N tuple data with the same teaching procedure. The results are shown in table 3 in the same format as before, some results are shown from the thresholded teach data for comparison.

It was suggested earlier that the use of non thresholded N tuple data during teaching caused saturation of the identifier memory and also produced a response pattern that prevented adequate occlusion analysis to take place. Because of this the results of this investigation are more difficult to interpret. Identifier recall success is low and there is no way to tell if the pattern recalled was the correct one. This is because all N tuple states are active in the output image and hence no outline of the shape is present; this prevents recognition of recall success by inspection. However by inspection of the results and comparison with the results of the previous run it was possible to determine that the class was recovered correctly in most cases where the identifier was recovered correctly. When the identifier was not recovered neither was the class pattern. This suggests that recall success in recognition could be determined by seeing if the identifier

was correctly recoved, thus allowing some analysis to be done on this data.

By comparing the results of the two investigations it is obvious that the use of the thresholded N tupled input pattern during teaching resulted in a much better recall success (also using confidences from the N tuple pattern in the test), which agrees with the conclusions of the previous investigation.

By considering the confidence levels for worst case class recovery it can be seen that the results are constantly lower than in the previous investigation. This would be expected because the saturated memory (caused by teaching a non thresholded picture) would yield a greater `background` response on each test. This is also shown by the results given in the `Average of class threshold when output is correct`.

In conclusion, the greater saturation of the memories caused by teaching non thresholded N tuple data is not offset by the benefits provided by teaching non thresholded N tuple data (the teaching of non thresholded N tuple data was thought to allow greater distortion in the teach patterns). Generalisation abilities are better served by having a thresholded teach process resulting in less saturation and easier occlusion analysis, then testing on a non thresholded input image which would result in a greater proportion of N tuples to be available to the recognition process and thus result in improved recognition.

Although this last statment has not been verified, the first set of results which used a thresholded teach image show no appreciable deficit in performance. When testing is done with or without a thresholded N tuple data. Further work needs to be done to check the exact validity of this claim.

12.7. Investigation 3 : Into the effects of teaching many patterns into the system

The final investigation examined the effect of recognition performance for increasing number of patterns taught into the system.

12.7.1. Teach Procedure

For this investigation the system was taught on thresholded N tuple data in one case and non thresholded N tuple data in the other, as in investigation 2. 10 window positions were taught for each shape (square and triangle) as described previously. After this initial teaching a new pattern was selected to be taught, this is shown in opl (face), This was selected so that it would be appreciably different than the patterns taught earlier, forcing every window position to be taught at, because the difference between the input image and the pre-teach test image would be greater than the threshold x.

Procedure :-

1) Provide a new identifier (different than previously taught).

2) Teach 4 window positions on the new pattern.

3) Test recall abilities (see below).

4) Goto 2 until halted.

12.7.2. Test Procedure

The test process consisted of testing five repeatable window positions on the image of a square and then five on the image of the triangle after each teach cycle. In this investigation the effect on the class confidences in the P->C recall process and of the accuracy of image

recall in the C -> P process are of interest. The following results were recorded, the results are given in table 4a - recognising a square, and in table 4b - recognising a triangle.

P->C test

i = Average of the response of points in the class above threshold.

ii = Average of response points in the class pattern below threshold.

iii = Worst case class points distance.

C->P tset

i = Tuples active in the response pattern not present in the input pattern.

ii = Tuples active in the response pattern also present in the input pattern.

iii = Number of class points active after N point thresholding.

Because testing is done on the same image as used in teaching and at the same positions the C->P, test should result in no points difference between the response pattern and the input pattern if recognition recall is perfect.

The values for the class response are a factor of 100 greater than actual and should be read by dividing by 100 to obtain the real results, this is due to integer arithmetic used throughout the simulations.

All tests were done with a thresholded input N tuple image and N tuple confidences were provided to the memory.

12.7.3. The Effects on Class Recall of Teaching Many Patterns into the System

The following refers to the results for the P->C tests given in table 4a and 4b. The main conclusion is that the response of class points above threshold remains constant as more images are taught into the memory whilst points below threshold rise with teaching. This confirms expectations from the analysis of memory properties (chap 7). As the memory becomes more saturated it causes more spurious outputs from the memory due to coincidences with N tuples taught on different class patterns. N tuples taught on the correct class for that pattern have all links formed thus further teaching on the same pattern cannot raise the response any further. The responses are not purely integer because confidences fed from the N tuples raise each link response of 1 by a factor given by the confidence.

The worst case class points distance rises in jumps as more patterns are taught . This is expected since data taught into the memory by the new pattern is randomly distributed and has a probability of effecting or not effecting a N tuple in the present input image.

The responses for the square are on average higher than those for the triangle and indicate that the the square has more active N tuples in its input pattern after teaching and/or the associated confidences of each N tuple are higher.

12.7.4. The Effect on Picture Recall of Teaching Many Patterns in to the System

The following refers to the results for the C->P test given in table 4a and 4b. The first point is that the class was always recovered with four points active indicating a high probability of a correctly

recovered class.

The number of N tuples active in the response and also present in the input image remains constant indicating a correct recovery on every test. The number of points in the response that are not in the input image after each test is above zero in most cases, indicating saturation occurring in the C -> P memory, even though only 24 patterns have been taught (6 sets of results, 4 patterns taught successively for each). The number of these points raises as teaching progresses as would be expected. Although the average error is quite low after 6 teach cycles (a total of 24 patterns taught) the results for individual tests vary widely, from a difference of 20% error between input and output patterns, to a perfect recall. Because of this wide variation in results no general conclusions can be gained.

12.8. Summary

To summerise, the following points can be made.

a) Teaching should be done using thresholded N tuple data. Thresholding consisting of taking the average confidence of all the N tuples in the N tupled input pattern and deleting all N tuples active in the input pattern whose confidence is less this average confidance. (Investigation 2)

b) Testing should involve using the confidences produced in the N tuple process to increase the effect of ´sure´ N tuples agains the ´unsure´ N tuples. (Investigation 2)

c) Testing should take place on non thresholded confidence tuples although no difference in reliability of the memory was found when thresholding and not thresholding was used. Theory suggests it should

improve recognition of a greater generalised set of patterns. It has the added effect of increasing processing speed during testing. (Investigation 2)

d) Re-testing the response picture to check the class patterns is ineffective at detecting recall error. (Investigation 2)

e) Occlusion analysis is more straight forward if thresholding the input N tuple pattern is performed. (Investigation 2)

f) Saturation of the identifier store is accelerated if non thresholded N tuples are taught. (Investigation 1)

g) Progressive teaching results in increased below threshold values in the class response image, with the effect that noise margins are reduced, and also increased probability of error in picture recall. (Investigation 1)

The results also indicate that recognition of occluded and occluding objects is possible using the system.

CHAPTER 13

Representation and Processing of 3 Dimensional Scenes of Occluded Objects.

13. Introduction

This chapter describes the process of occlusion analysis described in chapter 4 in relation to the memory and input processing described earlier. It covers the representation used to describe the 3 dimensional relationship between shapes present in a 3D scene and discusses the use of the high resolution window to disambiguate objects which cannot be classified using the low resolution window data.

The following attributes of the system are discussed.

1) Detection of correct recall of the patterns from the memory.

2) Formation of a complete description of a shape, after recognition over a number of window positions.

3) Production of a 3 dimensional description of a scene.

4) Use of the high resolution window to resolve ambiguity, and integration of high resolution data with the low resolution data.

13.1. Representation of Object Descriptions after Recognition

As each object is recognised in an image it's individual shape is stored in a data structure which allows the 3 dimensional relations between shapes to be analysed and described.

In this system, each shape recognised is allocated a storage array.

Data within each array describes the N tuple data for each object in the scene.

Because a unique identifier is available for each shape recognised, this is used to index the storage planes associated with each object. To make this process as simple as possible each identifier pattern is a pattern where only one element is set to 1, all others are set to zero. Each element of the output identifier pattern points to one storage plane. As shown in Fig 13.1. Each storage plane consists of a set of locations exactly in the form of the N tuple response image but now each element is non binary, i.e can store values greater than one.

An object description is built in these planes using the response from the recognition system in the following way.

Upon each test, an identifier image and a response image is produced. The identifier pattern indexes one storage plane, in which the response image can be added.

Each successive test produces an identifier image which indexes further planes in the storage arrays to which response images are added. The position the images are added in the storage planes is

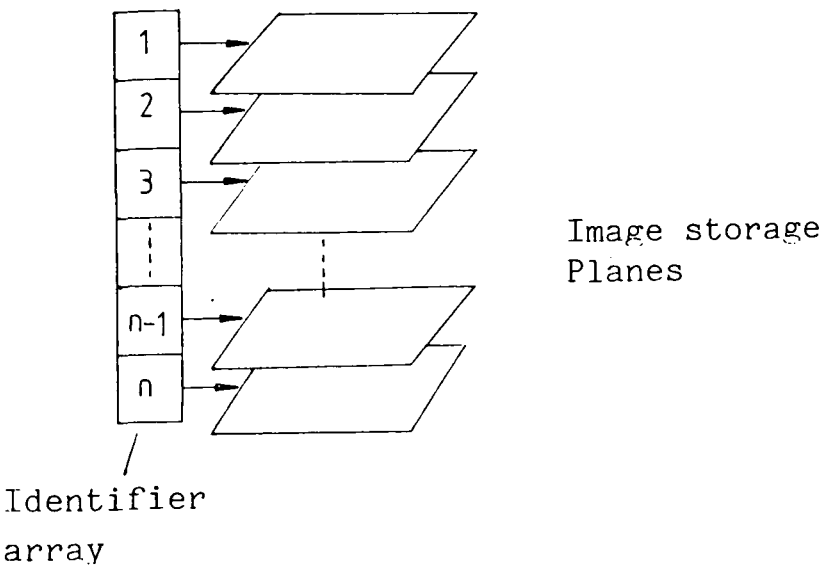


Fig 13.1
Image storage planes

relative to the scene i.e the array in which the responses from the memory occur in, is placed in the image array at a point relative to the position of the input window in the input array. As each pattern occurs in the output from the memory it reinforces images previously added into the storage planes. Furthermore the images generated in the storage planes are in registration with the objects in the input image, this is necessary for occlusion analysis as will be explained later.

Before adding a shape to the storage planes each response image is first checked to see if it is a valid image by checking whether it was generated by an N point class. If it was the image is added. Other tests to check for correct recall are be done as explained in chapters 11 and 12, such as the response image feedback process, but as explained in chapter 12, it is impossible to reliably check whether the recalled image is correct or not. The process of adding the response image to the storage planes is used to build up a reliable image of what the system recognises in the input image. Incorrect recognition of shapes would constitute a small number of the total number images recalled from the memory, addition of these images to the storage planes effectively adds noise to the the storage planes. The correctly recovered shapes reinforce each other, whilst incorrect images added remain as background noise within the storage arrays. After recall and addition of a number of images into the storage arrays, the arrays are thresholded to recover the binary tuple patterns of the objects recognised. Fig 13.2 illustrates a triangle with `X`s marking the four positions visited in the image, out of these 4,3 and 2 had been previously visited and taught. Position 1 has not been taught on the triangle. The system is also assumed to have been taught on a square.

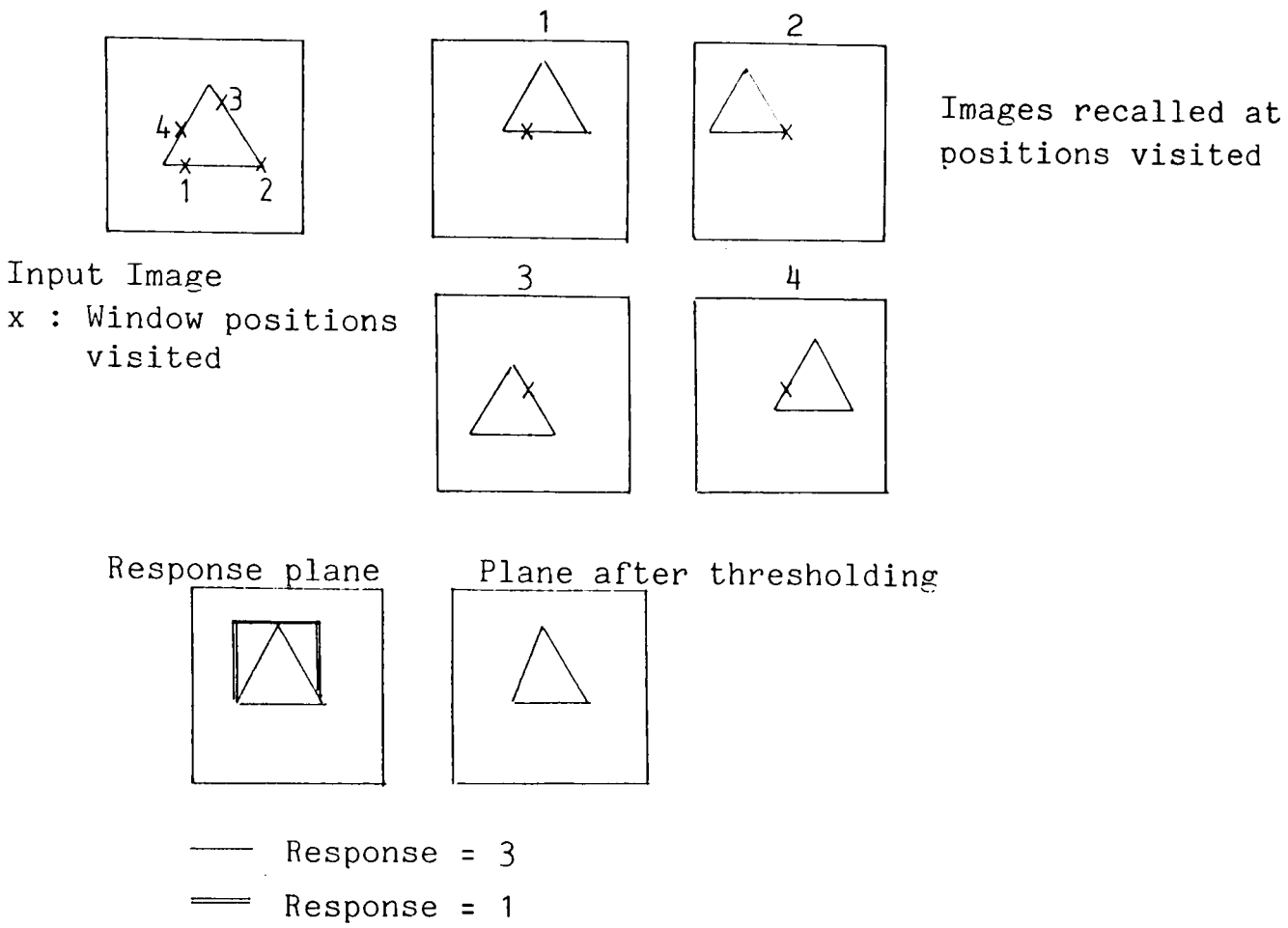


Fig 13.2
Building a consistent representation of a shape.

The response images show the correct responses at the positions 4,3, and 2. At position 1 a square is incorrectly recovered. These images are assumed to generate the same identifier pattern thus the same storage plane is indexed on each recall. The storage plane is shown with the images recalled added into it. The three correct images recalled prevail over the incorrect image which, after thresholding, is reduced to a binary tupled image of the correct object.

Because each image plane contains information pertaining to one object in the scene, and that that object position in the image plane is relative to the position of the object in the input image, the image planes are termed `Object Based frames`. The term `frames` is derived from its use in AI as defined by Minsky Minsky1975 only its use here is a simple form of the concept of frames.

The process of thresholding the data within the object frames remains

problematic. As with earlier work on thresholding the class response from the memory, the process of thresholding the object frames should not be based upon an arbitrary selected value. Ideally some recall information should be used to find the ideal threshold so that the threshold is sensitive to variations in recall ability. At present no such process has been found, the image is thresholded at the average response of all the data points in each object frame.

The summing of recalled images into the object frames must be linked to some value indicating the measure of certainty that the recalled image is correct. For instance if a recalled image can be said to have a confidence of 0.5 of being correct then each value for each tuple in the recalled image can be multiplied by 0.5 and added to the relevant object frame. Those recalled images which have little likely hood of being correct do not overly effect the values so far summed into the object frame. The confidence that the recalled image is correct is derived from the class response data on the P->C memory test in the following way. First the input image is tested in the normal way to produce a class response array. This is then thresholded at the set N point threshold. The difference between the highest responding non selected class point and the lowest responding selected class point is noted. This value is divided by the maximum possible class point response to derive a figure of confidence of correct recall.

The object frame process has not been fully implemented and thus remains a topic for further work. Initial simulations show recall inaccuracies in the positioning of the window in the input image, this is due to the coarseness of the edge detectors and results in a blurred image of the shape in the object frames. It is not yet clear whether this will effect the occlusion analysis process described in chapter 4.

13.2. Recognition of all Objects within a Scene

The recognition system recognises the `largest` object in the image first, that is objects which contain the largest number of N tuples. This was explained in chapter 6, it concerns the case shown in Fig 13.3 where two shapes share the same boundaries. In this example two squares are shown, the small square sharing two of its edges with a part of the large square. If the system were to window on point p (and assuming the system has seen this view of the two shapes previously) the system would respond with the large square in preference to the small.

If the system were to window on the interesting point marked i, only the small square would be recognised. This is because the large square would not have been taught at these positions, since no interesting areas would have occurred at these points. Recognition of the small square can also be aided by the use of the high resolution window since it would restrict the systems view to that of the small square.

The system slowly builds up views of each object in the object frames until thresholding shows that most input image points belong to some shape. Areas of the image that are not represented in the object frames are selectively centred upon to find their identity.

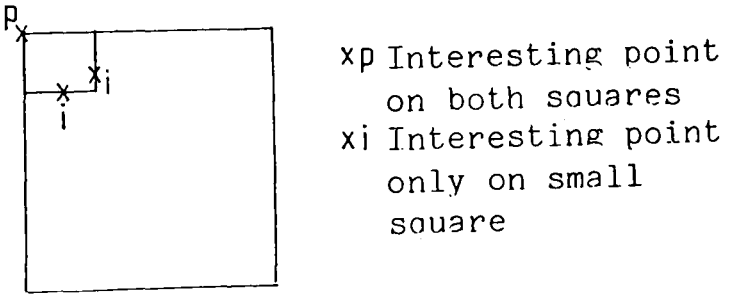


Fig 13.3
Recognition of two squares.

13.3. Analysis

After the whole scene has been parsed for objects the system analyses the scene for occlusion information. The method described in chapter 4 is applied to the data in the object based frames. Briefly, this method starts by identitying candidate occluding features by comparing all object frames with each other, then following on to analyse each feature found for the occluding and occluded feature. Each object frame is then be labelled to indicate which object the shape represented within is occluding (if any).

13.4. Use of High Resolution Data

To enable separation of very similar images, the high resolution window is used. To show how this is used, consider Fig 13.4. This shows two images which are to be taught and recognised. The two images differ by only a small amount, i.e. the small section in the top right hand corner. Let us assume that the image A has been taught under the identifier shown at four window positions, these being the corners of the shape A. Each is taught under a separate class pattern randomly selected.

Shape B is now presented for testing. The process starts by identifying positions of interest, i.e. the four corners of the shape B. The

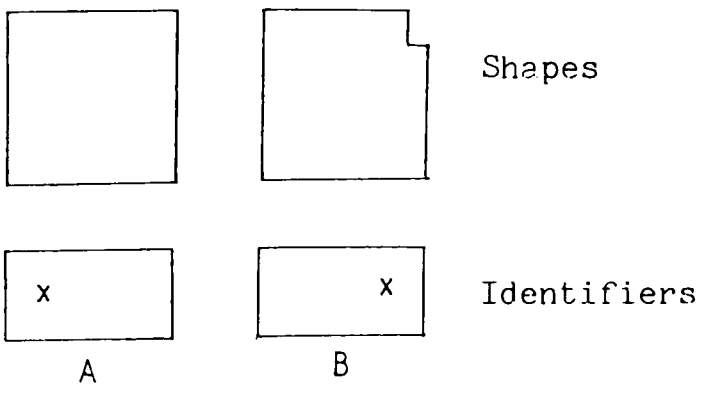


Fig 13.4
Separation of two similar shapes.

first point is selected, this could be the bottom left hand corner, the window centred at this point and the memory tested. This results in the responses from the system shown in Fig 13.5. The class and recovered image are recalled along with the identifier for shape A. Comparing the input with the response shows a difference occurring at the point indicated. As explained previously, if the difference between the input image and the response is less than threshold, teaching cannot take place in the low resolution window in the normal way as this would result in poor recovery during testing due to reasons discussed previously.

The high resolution window is now used to separate shape A from shape B. Before going on to teach the high resolution window it is necessary to indicate that the recognition of shape A cannot be done wholly in the low resolution window. To do this the following teach process is used,

1) The input image is taught under the current class only into the P-

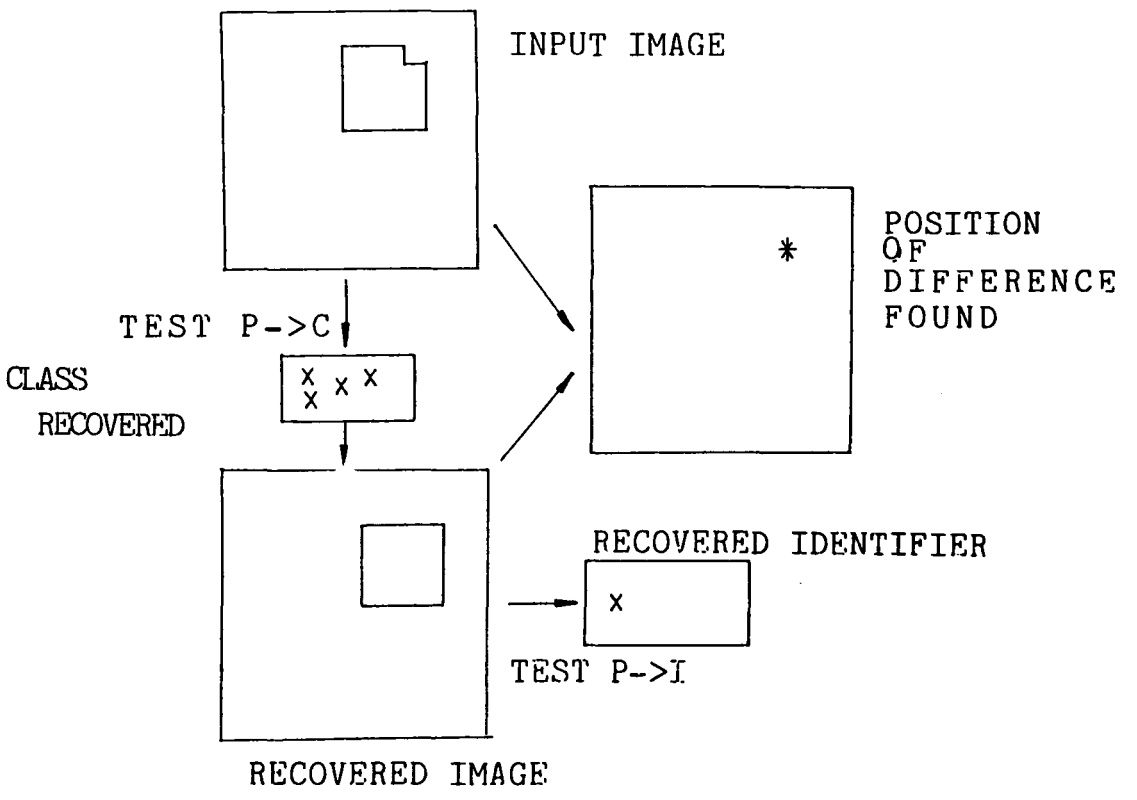


Fig 13.5
Identification of positions of interest.

>C memory - this results in generalisation between patterns A and B under the current class, thus both A and B will respond with this class on subsequent tests.

2) The C->P memory is taught on the input image and the recovered class - this results in generalisation between A and B images for recognition. This image will then be used to identify high resolution points visited as explained below.

3) A new identifier is taught under the input pattern in to the identifier memory - This results in a generalisation between the two identifiers for shapes A and B. Testing will result in the output of both identifiers.

Before going on to teach the high resolution data, it is necessary to visit all other points of interest in the low resolution window at which shape A was taught, at each point the above teaching should be performed to prevent the system recognising shape shape A at any window position.

The above low resolution process obliterates the ability to recall image A and its identifier. Testing along the lines discussed at the beginning of the chapter will result in two object frames holding exactly the same image. This is because testing at any positions in A or B will produce an identifier pattern which will index two object frames. Both frames will be updated accordingly with the generalised patterns of A and B.

The output during testing is shown in Fig 13.6. Most N tuple data in the object frames would be singular, i.e only one state active in any one N tuple. But, at the point indicated in the diagram, a confusion would exist where two N tuple states within one N tuple would be

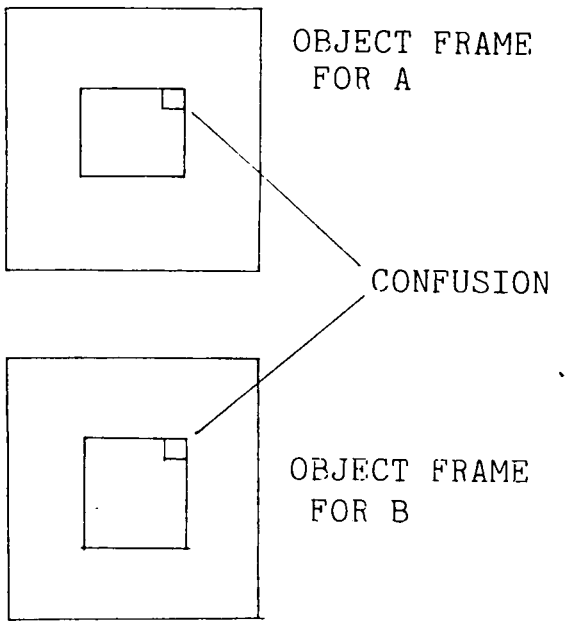


Fig 13.6
Confusion in object frames.

equally active. This indicates to the recognition system that confusion exists and that high resolution windowing is needed at the point where confusion exists.

Thus, the low resolution window has been taught to indicate to the testing process that recognition confusion exists.

The high resolution window teaching is now performed. At one point visited in the low resolution window, the area of difference can be identified between the input image and the response image and move the high resolution window to this position. The system moves on to detect the point of interest in the high resolution window, as done in the low resolution window. At each point found the the pre-test process is repeated and a difference test done. If a difference results in a value greater than threshold teaching takes place. The system teaches a new class and identifier for shape B in both the high and low resolution memories. If a difference is not found to be greater than threshold then the system teaches just the inputed image on the recovered class into both P->C and C->P memories and teaches a new identifier under the inputed image to produce similar confusion in

recall as in the low resolution case explained above.

The result of the above process is to allow recognition of shape B during testing. The high resolution image recovered on each test is disguarded and the identifier recoved used to index the object frames for storage of the low resolution image.

Disguarding the high resolution image data is perhaps a waste of storage. If small objects were to be recognised or detail in larger objects needed to be examined, this information would be essential. To allow this the resolution of the object frames is altered - to allow storage of detailed image information.

The above process resulted in the ability to recognise shape B by access to high resolution data, but an inability to recognise shape A. This resulted because high resolution information is not available to recognise shape A particularly. To enable this shape to be recognised it is re-presented to the system and the system allowed to learn the high resolution information needed to recognise it. Some form of inference mechanism may be used to indicate if both A and B were equally recognisable in low resolution and just B was NOT recognised in high resolution then the input object could be classified as A.

In the case illustrated, the difference between image A and B could be discriminated in low resolution. It is quite possible that two shapes are different but no indication would be present in low resolution. This makes the above process ineffective. The difficulty in recognising such objects is great. Scanning the high resolution image over the scene could be performed when such differences cannot be recognised in low resolution, but this causes problems in recognition, i.e. the inability of the system to identify where the high resolution window is to be placed. This necessitates the use of a storage process involved

in memorising the positions that the high resolution window was taught at. This is left for further work.

13.5. Summary

This chapter has described how objects recognised within a complex 3D scene may be stored in a number of object frames. These frames are indexed by the identifier pattern which is recalled along with each pattern. Upon recognition each pattern is added to an object plane, which may be subsequently thresholded to recover the prevailing patterns in the scene. Occlusion analysis as described in chapter 4 is then applied to the data contained in the object planes to discover the relative depth of objects within the input scene.

CHAPTER 14

Conclusions and Further Work

1. Conclusions

This thesis has investigated a novel scene analysis system which is capable of recognising occluded objects within a three dimensional scene. A method of discovering the relative positions of shapes has been proposed, along with a novel representation of this information. Briefly this is based upon the associative recall of a complete shape from its incomplete input image. After all shapes have been recognised and stored in object frames a simple occlusion analysis process is performed. To enable this process of occlusion analysis to operate an associative memory has been developed which has been shown to be effective at recognising and recalling complete descriptions of objects in simple scenes. The associative memory has been based upon a simple mapping memory. It is a two stage device, where binary storage elements have been used to give the memory a simple implementation and efficient storage ability. A formal analysis of the storage properties and recall abilities of the associative

memory has been given, and it has been shown to operate efficiently in terms of storage, accuracy and recall ability. In comparison with other mapping associative memories this new device shows a number of distinct advantages. These relate to the control of storage ability, the method and accuracy of recall, and the memories pattern recognition abilities. The use of a 2 stage memory is unique and has allowed a reduction in storage. The use of an intermediate class, pattern and the N point thresholding process enables recall to be made in a fast and accurate manner (previous methods discussed in chapter 5 were either slow or inaccurate). The new memory allows greater flexibility in design and using the formulae described in chapter 7, the storage ability and error rates may be accurately set. The memory offers a fail soft ability which introduces a high degee of reliability into the memory. The memory has many other applications outside the area of the present study, i.e in the area of speech recognition and large database design.

The N tuple preprocessing stage which is added to the associative memory allows for flexible storage control, however its main application is to enhance the pattern recognition abilities of the memory. A thorough analysis has been given to the N tuple

process in the context of occlusion analysis and has resulted in a new and efficient mapping of N tuple samples over a number of windows of different resolutions. Furthermore, it has meant the design of a grey scale encoding process. This allows direct encoding of the grey scale landscape seen by one N tuple as one of a particular set of binary states. This technique has been evaluated in comparison with the normal binary N tuple technique and has shown distinct advantages, manifested as increased recall success from the associative memory. This technique may be applied to other areas where the recognition of grey scale images is required.

To enable the recognition of both block filled and line drawn images an edge operator pre-processing stage has been implemented. This, in conjunction with a fovial multi window input system, has allowed a certain amount of similarity with the human visual system to be introduced, the implementation involves a simple Roberts style operater which has been shown to be sufficient in this application. Later work may use this implementation to investigate the reasons why the human visual system incorporates these facets. In the present context multiple concentric windows of different resolutions have allowed a reduction

of processing and storage requirements leading to lower cost and higher processing speeds.

The use of a fovial window input stage has meant the development of various high level recognition strategies to resolve ambiguity when dealing with recognition a number of different resolutions. It has been shown how a high resolution window may be used to remove confusion when an object is not recognisible in low resolution.

The system has been shown to be effective at recognising simple 3D scenes containing 2D objects. The representation of the objects after recognition in the form of 'object frames' has provided the possibility of applying the occlusion analysis processes proposed, as it provides a complete description of all objects present in their relative 2D positions.

It has been shown that a radical approach to the problem of scene analysis by the design of inherently parallel algorithms results in a system which may be implemented directly in hardware. It must be noted that the associative memory was simulated at a level which is very close to one possible method of dedicated implementation, thus the evaluation of this system effectively

has also evaluated this implementation.

1.1. Further Work

Further work remains to be done on many aspects of the system. Work needs to be done on predicting the memory storage abilities in the case where the input pattern contains noise during recall, thus allowing a far more accurate prediction of the performance of the memory. The effects of using more than two resolutions of input windows needs to be investigated along with the use of more edge detectors and different N tuple sample sizes and mappings. The realisation of the inherent hardware model needs to be investigated - this would allow proceesing of complex scenes within one second.

Throughout the thesis the model proposed has been compared to the human visual system. A more detailed investigation is needed that examines the similarity of the system with the known neurophysiology of the brain. This would both help the understanding of the brain and perhaps lead to the solution of many remaining problems in the model.

While the associative memory is ideal for the recognition of data in the spatial dimention, it may be easily extended for to be able to analyse temporal data by the addition of pattern feedback (output -> input). This may allow the syntactic analysis of complex scenes as well as motion analysis.

The control alogarithms, which are used to sequence the analysis of scenes may be developed much further. Particularly in to merging recognition with teaching as suggested in the text. The alogarithm may also incorporate the distinction between passive scene analysis (i.e "Whats in the scene?") and active scene analysis (i.e "Where is the cube ?").

At present the system only deals with rotation and scale invarient recognition by a brute force approach (i.e teaching all views). Further work needs to be done on this aspect of the system, perhaps using parallel devices with similar structure as the associative memory used in the system.

REFERENCES

[Boden1977]
M. Boden, Artificial Intelligence And Natural Man, Harvester Press, 1977.

[Brady1981]
M. Brady, "The Changing Shape Of Computer Vision," Artificial Intelligence, vol. 17, pp. 1-15, 1981.

[Guzman1968]
A. Guzman, "Decomposition Of A Visual Scene Into Three-dimentional Bodies," Fall Joint Computer Conference, pp. 291-304, 1968.

[Shirai1973]
Y. Shirai, "A Context Sensitive Line Finder For Recognition Of Polyhedra," Artificial Inteligence, vol. 4, pp. 95-120, 1973.

[Marr1982]
D. Marr, Vision, W. H. Freeman And Co., 1982.

[Gonzalez1978]
R. C. Gonzalez and M. G. Thomson, Syntactic Pattern Recognition: An Introduction, Addison-Wesley Pub. Co., 1978.

[Ullmann1983]
J. R. Ullmann, "An Investigation Of Occlusion In One Dimension," Computer Vision, Graphics, And Image Processing, vol. 22, pp. 194-203, 1983.

[Aleksander1985]
I. Aleksander and M. J. D. Wilson, Adaptive Windows For Image Processing, 1985.

[Stonham1985]
J Stonham, in Advanced Digital Information Systems, edited by I. Aleksander, pp. 231-272, Prentice Hall International, 1985.

[Rosenblat1958]
F. Rosenblat, "The Perceptron A Probablistic Model For Information Storage And Organisation In The Brain.," Psyc. Rev., vol. 65, pp. 386-408, 1958.

[Culloch1943]
W. S. Mc Culloch and W. Pitts, "A Logical Calculus Of The Ideas Imminent In Nervous Activity," Bulletin Of Mathematical Biophysics, vol. 5, pp. 115 - 133, 1943.

[Minsky1969]
M Minsky and S. Papert, Perceptrons, MIT Press, 1969.

[Pinker1985]
S. Pinker, Visual Cognition:An Inroduction, 1985.

[Turney1985]
J.L. Turney, T. N. Mudge, and R. A. Volz, "Recognizing Partially Occluded Parts," IEEE Trans. Pattern Analysis And Machine Intelligence, vol. PAMI-7, 4, 1985.

[Berman1985]
S. Berman, P. Parikh, and C. S. G. Lee, "Computer Recognition Of Two Overlapping Parts Using A Single Camera," IEEE Computer, vol. 18, pp. 70-81, 3, 1985.

[Dawson1976]
C. Dawson, "Aspects Of Simple Scene Analysis With Learning Networks," PhD Thisis University Of Kent, 1976.

[Brady1983]
M. Brady, "Parallelism In Vision," Artificial Intelligence, vol. 21, pp. 271-283, 1983.

[Bledsoe1959]
W. W. Bledsoe and I. Browning, "Pattern Recognition And Reading By Machine," Proc. Joint Comp. Conference, pp. 255-232, 1959.

[Bledsoe1961]
W. W. Bledsoe, "Further Results On The N Tuple Pattern Recognition Method," IRE Trans. Comp., vol. EC-10, p. 96, 1961.

[Bledsoe1962]
W. W. Bledsoe and C. L. Bisson, "Improved Memory Matricies For The N Tuple Pattern Recognition Method," IRE Trans Comp., vol. EC-11, pp. 414-415, 1962.

[Highleyman1960]
W. H. Highleyman and L. A. Kamentsky, "Commensts On A Character Recognition Method Of Bledsoe And Browning," IRE Trans. Comp., vol. EC-9, 1960.

[Stonham1986]
T. J. Stonham, "Aspects Of Face Processing," in To be pub, Martins Nighoff, 1986.

[Wilkie1983]
B. A. Wilkie, "A Stand Alone High Resolution Pattern Recognition System," PhD Thesis Brunel Universisty, 1983.

[Aleksander1984]
I. Aleksander, W. V. Thomas, and P. A. Bowden, "WISARD:A Radical Step Forward In Image Recognition," Sensor Review, pp. 120-124, July, 1984.

[Kohonen1977]
T. Kohonen, Associative Memories: A System Theoretical Approach, Springer-Verlag, Berlin, 1977.

[Ullman1969]
J. R. Ullman, "Experiments With The N Tuple Method Of Pattern Recognition," IEEE Computors, vol. C-18, pp. 1135-1137, 1969.

[Aleksander1983]
I. Aleksander, "Emergent Intelligent Properties Of Progressively Structured Pattern Recognition Nets," British Pattern Recognition Association, Second International Conference, 1983.

[Waltz1975]
D. A. Waltz, "Understanding Line Drawings Of Scenes With Shadows," in The Psychology Of Computer Vision, edited by P. H. Winston, pp. 19-92, Mc-Graw Hill, 1975.

[Zissors1983]
Zissors, Colour Encoding Interface For A Pattern Recognition Divice, MSc. Brunel University, Dept Electrical Eng., 1983.

[Stonham1974]
T. J. Stonham and I. Aleksander, "Optimisation Of Digital Learning Networks When Applied To Pattern Recognition Of Mass Spectra," Electronics Letters, vol. 10, pp. 301-302, 15, 1974.

[Patel1986]
M. Patel, "Optimisation Of N Tuple Networks," PhD: Brunel University, Dept. Electrical Eng., 1986.

[Ullman1971]
J. R. Ullman, "Reduction Of The Storage Requirements Of Bledsoe And Brownings N-tuple Method Of Pattern Recognition," Pattern Recognition, vol. 3, pp. 297-306, 1971.

[Steck1962]
G. P. Steck, "Stochastic Model For The Browning-Bledsoe Pattern Recognition Scheme," IRE Trans. Elect. Comp., vol. EC-11, pp. 274-282, 1962.

[Roy1967]
B. J. Roy and J. S. Sherman, "Two Viewpoints Of k-tuple Pattern Recognition," IEEE Trans. SSC, vol. SSC-3, pp. 117-120, 2, 1967.

[Aleksander1985.....]
I. Aleksander, "Memory Networks For Practical Vision Systems," Pattern Recognition Letters, vol. 1:3, 1985.

[Torre1986]
V Torre and T A Poggio, "On Edge Detection," IEEE Pattern Analysis And Macine Inteligence, vol. PAMI-8 No 2, 1986.

[Wilson1985]
M. J. D. Wilson, "Adaptive Windows:Edges,Steriopsis And Stripes.," British Pattern Recognition Association Third Int. Conf., 1985.

[Palm1980]
G. Palm, "On Associative Memory," Biological Cybernetics, vol. 36, pp. 19-31, 1980.

[Kohonen1984]
T. Kohonen, Self-Organisation And Associative Memory., Springer-Verlag, 1984.

[Lee1985]
R. M. Lee, in Advanced Digital Information Systems, edited by I. Aleksander, Prentice Hall International, 1985.

[Gardener-Medwin1976]
A. R. Gardener-Medwin, "The Recall Of Events Through The Learing Of Associations Between Their Parts," Proc. R. Soc. Lond. D., vol. 194, pp. 375-402, 1976.

[Willshaw1969]
D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-Holographic Associative Memory," Nature, vol. 222, pp. 960-962, June 7, 1969.

[Heerden1970]
P. J. Van Heerden, "Models For The Brain," Nature, vol. 225, pp. 177-178, January 10, 1970.

[Ballard1983]
D. H. Ballard, G. E. Hinton, and T. J. Sejnowski, "Parallel Visual Computation," Nature, vol. 306, pp. 21-26, 3, 1983.

[Lansner1985]
A. Lansner and O. Ekeberg, "Reliability And Speed Of Recall In An Associative Network," IEEE Tarns. Pattern Analysis And Machine Intelligence, vol. PAMI-7, 4, 1985.

[Palm1984]
G. Palm and T. Bonhoeffer, "Parallel Processing For Associative And Neuronal Networks," Biol. Cybernetics, vol. 51, pp. 201-204, 1984.

[Sivilotti1985]
M Sivilotti, M Emery, and C Mead, "A Novel Associative Memory Implemented Using Collective Computation," 1985 Chapel Hill Conference On VLSI, pp. 329-342, 1985.

[Hopfield1982]
J. J. Hopfield, "Neural Networks And Physical Systems With Emergent Collective Computational Abilities," Pro. Natl. Acad. Sci. USA, vol. 79, pp. 2554-2558, 1982.

[Wright1983]
M J Wright and A Johnson, "Spatio Temporal Contrast Sensitivity And Visual Field Locus," Vision Res., pp. 983-989, 1983.

[Davis1976]
L S Davis, A Rosenfeld, and A K Agrawala, "On Models For Line Detection," IEEE Trans. Syst. Man And Cyber., vol. SMC-6 No 2, 1976.

[Davis1975]
L S Davis, "A Survey Of Edge Detection Techniques," Computor Graphics And Image Processing, vol. 4, 1975.

[Yarbus1967]
A L Yarbus, Eye Movements And Vision, Plenum Press New York, 1967.

[Makworth1967]
N H Makworth and A J Morandi, Perception And Psychophysics, vol. 2 Mo 11, 1967.

[Moravec1977]
H. P. Moravec, "Towards Automatic Visual Obstacle Avoidance," Proceedings Of IJCAI 77, vol. Vision-1, p. 584, 1977.

[Minsky1975]
M Minsky, "A Framework For Representing Knowledge," in The Psycology Of Computer Vision, edited by P. H. Winston, Mc-Graw Hill, 1975.

1. APPENDIX 1 : Derivation Memory Saturation Formula

Derivation of the equation to give the number of bits set in the memory after T teach iterations used in chapter 7.

Label conventions

R = Number of rows in memory matrix

NI = Number of rows selected in R on every teach/test

H = Number of columns in memory matrix

N = Number of columns selected in H on every teach/test

The memory consists of a matrix of points R by H in size. Each time a pattern or pair of patterns is taught into the memory it becomes filled by a certain amount. This is represented by the number of bits set in the matrix.

To simplify consider the case where R = 1 (H > 1). Thus we have one a one dimensional matrix of H elements. On each teach we place N points into the array at random. After a number of trials a certain number of locations will be set in the array. This will be derived in the following.

let S_t = Number of elements set after t teaches

P_t = Number of elements set in trial t + 1

$$P_t = N \times \frac{H - S_{t-1}}{H} \tag{1}$$

$$S_t = S_{t-1} + P_t \tag{2}$$

so after the first teach ...

$$P_1 = N$$

$$S_1 = N$$

Then after the second ...

$$P_2 = N \times \frac{H-N}{H} \qquad \text{From (1)}$$

$$= N - \frac{N^2}{H}$$

$$S_2 = N + N - \frac{N^2}{H} \qquad \text{From (2)}$$

$$= 2.N - \frac{N^2}{H}$$

After the 3'ed ...

$$P_3 = N \times \frac{H - S_2}{H}$$

$$= N \left(1 - \frac{2.N}{H} + \frac{N^2}{H^2}\right)$$

$$S_3 = S_2 + P_3$$

$$= 3N - 3\frac{N^2}{H} + \frac{N^3}{H^2}$$

After the forth ...

$$P_3 = N - 3\frac{N^2}{H} + 3\frac{N^3}{H^2} - \frac{N^4}{H^3}$$

$$S^4 = 4N - 6\frac{N^2}{H} + 4\frac{N^2}{H^2} - \frac{N^4}{H^3}$$

It can be seen that S_t is progressing as a series with binomial coefficient.

$$S_t = \sum_{r=1}^{t} C_r^t \, (-1)^{r-1} \cdot (N / H)^{r-1} \cdot N$$

So now we have a formula for a one dimensional matrix that gives the number of points set after t teaches.

This now needs to be extended to a 2D matrix. The equation for P_t is now

$$P_t = N \cdot NI \cdot \frac{H \cdot R - S_{t-1}}{H \cdot R}$$

Which follows through to make

$$S_t = \sum_{r-1}^{t} c_r^t \, (-1)^{r-1} \cdot (N \cdot NI/H \cdot R)^{r-1} \cdot N \cdot NI$$

This equation can be reduced to a polynomial and simplified,

$$S_t = H.R \, (1 - (1 - (N.NI/H.R)^{t}))^{r}$$

2. APPENDIX 2 Derivation of Formulae in Chapter 9

The following gives the derivation of the formula in chapter 9 to give the 'number of patterns that can be taught before the Hamming distance between a taught and a tested pattern, averaged over a number of tests, becomes greater than one' following the conventions used in Chapter 7.

The probability of selecting a point in the memory matrix that is 1, after T patterns have been taught is given by,

$$p(\text{active location}) = 1 - (1 - (N.NI/H.R))^{T}$$

From equation (3) chap 6

And so the probability of getting one error in one column can be expressed as,

$$p(\text{error column}) = p(\text{active location})^{NI}$$

$$= (1 - (1 - (N.NI/H.R))^{T})^{NI}$$

Now the probability of getting one and only one error in all columns of the memory matrix on a test is given by the second term of the binomial,

$$(p(\text{active location}) + p(\text{not active location}))^{H}$$

~~expanded gives,~~ which is

$$e(1) = C_1^H \, p(\text{acive location})^1 \cdot (1 - p(\text{activelocation}))^{H-1}$$

Now $C_1^H = H$, so substituting,

$$e(1) = H(1 - (1 - (N.NI/H.R))^{T})^{NI} \cdot$$

$$(1 - (1 - (1 - (N.NI/H.R))^{T})^{NI})^{H-1}$$

This gives the probability of getting one and only one error out of all columns in the memory array on any one test. Thus represents the probability of getting one bit hamming distance on a test compared to that taught.

We now need to know when it is most likely , on average, to get a hamming distance of 1. So after, say, 100 patterns are taught, if 20 patterns were tested the average hamming distance would be 1.

This can be found, since the function for e(1) has a maximum, which represents the time at which you are most likely to get one error from memory. If the equation were to give the probability of getting two errors, the maximum would be at a different point.

Thus to obtain the maximum of the function e(1) we need to first differentiate e(1) with respect to T,

$$\frac{de}{dT} = (1 - (c + 1)(1 - a^{T})b^{c-1}) \cdot b(1 - a^{T})^{b-1} \cdot a^{T} \cdot \ln a$$

where

$$a = 1 - \frac{N.NI}{H.R}$$

$$b = NI$$

$$c = H - 1$$

Now find the maxmimum, when $\frac{de}{dt} = 0$

This reduces eventually to

$$T = \frac{\ln\left(1 - \left(1 \,/\, H^{(1/NI)}\right)\right)}{\ln\left(1 - (N.NI/H.R)\right)}$$

3. APPENDIX 3 Construction of Edge Operators

The physical realisation of a tuple formed from edge detectors is given here.

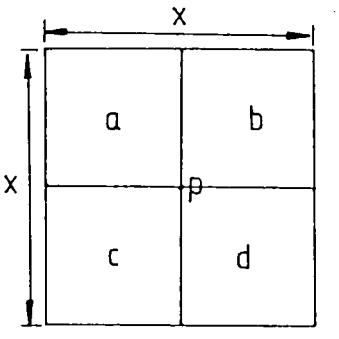


Fig A.1
Format of edge operator

The figure above shows the format of an edge operator with a central point p. The vertical line from p is given the classification 0^o to give a datum for edge operators. The four quadrants of the above region are labelled a - d each covering an area of $(x/2)^2$ pixels in the scene. The equations for the edge operators 0^o and 90^o are

$$90^o = |(a + b) - (c + d)| + |(c + d) - (a + b)|$$

$$0^o = |(a + c) - (b + d)| + |(b + d) - (a + d)|$$

For the edge operators at 45^o and 135^o the above grid is rotated by 45^o and the same formulae used.

These four responses are then supplied to the tuple process described in chapter 3. The value of x differs for the high resolution and low resolution images. In all cases x = 16 pixels for low resolution image and x = 8 for high resolution images. Sampling with the N tuples is

done at x/2 intervals across the image.

3.1. Relative Sizes of Windows

The simulation presently has two windows, low and high resolution the relative sizes of these are low 1:2 (high resolution:low resolution).

4. APPENDIX 4 : System Parameters to Repeat Experiments in Chapter 12

This lists the various system parameters which would be needed to repeat the experiments reported in chapter 12.

Input image size ; 128 x 128 pixels, 256 grey scale.

Class image size ; 4 x 8 pixels

Points set to one in class pattern ; 4

Edge operator dimensions ; See appendix 3

Identifier Image ; 8 x 8 pixels, arbitrary shapes used to denote identity.

Interesting area detection using edge operator primitives.

Threshold of confidence tuples ; When used, threshold at average response of all confidences in present response image.

As well as these parameters a process of reducing the time calculating edge operator responses was used. By reducing the 16 x 16 values collected per edge operator by a factor of x, named the mapping factor. In all experiments x was set to 4. So only 1 in 4 samples of 16 x 16 possible edge operator samples were actually collected.

5. APPENDIX 5 : Graphs

The following describes the graphs shown on the following pages.

Graph A_2

This gives the predicted probability of error for the `picture to class memory´ after a number of patterns have been taught. The number of class points set to 1 is varied for each plot.

Picture array size = 896

Number of elements in the picture set to 1 = 64

Class array size = 128

Number of class points set to 1 : A=8 B=15 C=30

Graph B_2

As for graph A_2, but varying the size of the picture array in each plot.

Picture array size: A=896 B=1792 C=2688

Number of elements in the picture set to 1 = 64

Class array size = 128

Number of class points set to 1 = 15

Graph C_2

As for A_2 and B_2, but varying the number of points in the picture set to one in each plot.

Picture array size = 896

Number of elements in the picture set to 1: A=64 B=40 C=20

Class array size = 128

Number of class points set to 1 = 15

Graph E_2

This graph gives the number of patterns that can be stored in the class to picture memory before the error rate raises above `x` (given below), over a range of class points set to 1. Each plot is for a different value of `x`.

x (probability of error threshold): A=0.1 B=0.01 C=0.001

Picture array size = 896

Number of points set to 1 in the picture = 64

The class array size = 32

Graph F_1

The number of images that can be taught into the picture to class memory for different numbers of class points set to 1, for a given probability of error (0.001). Different plots are for different class array sizes.

Picture array size = 896

Number of points set to 1 in the picture = 64

Probability of error = 0.001

Class array sizes : A=32 B=64 C=128

Graph F_2

As in Graph F_1 but for the class to picture memory.

Picture array size = 896

Number of points set to 1 in the picture = 64

Probability of error = 0.001

Class array sizes : A=32 B=64 C=128

Graph G_1

See chapter 6 for explanation.

Probability of error : A=0.001 B=0.01 C=0.1

Picture array size = 896

Number of points set to 1 in picture = 64

Graph H_1

The number of images that can be stored before error = 0.001 as the class array is varied in size. Each plot is for different picture sizes.

Number of class points set to 1 = 5

Picture array size : A=1024 B=2048 C=4096

Number of picture points set to 1 : A=64 B=128 C=256

Graph H_2

As for graph H_1, but of much larger picture sizes.

Number of class points set to 1 = 5

Picture array size : A=16384 B=65536 C=262144

Number of picture points set to 1 : A=1024 B=4096 C=16384

Graph I_1

This contrasts two mesurments, first, plot A, the number of class points set to 1 needed to obtain optimal storage from the memory, for a given class array size. Second, plot B, the number of pattern associations that can be stored at the optimal number of class points set to 1. This relate to the class to picture memory.

Picture array size = 896

Points set to 1 in the picture = 64

Probability of error = 0.001

Graph I_2 This gives the number of class points set to 1 needed for optimal storage in the class to picture memory (it expands plot A in graph I_1).

Picture array size = 896

number of points set to 1 in the picture = 64

Probability of error in recall = 0.001

Graph J_1

This gives the saturation rate of the memory. Different plots are given for different numbers of class points set to 1.

Number of picture points set to 1 = 64

Size of the picture array = 896

Class array size = 64

The number of class points set to 1 : A=32 B=16 C=4

5.1. Simulation and Mathamatically Predicted Results Graphs

The following graphs compare the mathematically predicted results with those from simulations. The system set up for each is given in the text (chapter 6).

Graphs Y_1 Y_2 and Y_3

This gives the number of images that can be stored in the picture to class memory before the hamming distance between the recoverd and expected pattern is greater than 1.

Y_1 and Y_3

A = Simulation result.

B = Mathematically predicted.

Y_2

B = Simulation result.

A = Mathematically predicted.

Graphs Z_1 Z_2 and Z_3

These compare the number of images that can be stored over a set of class array sizes. Mathematical and simulation results are compared.

Z_1, Z_2 and Z_3

A = Mathematically predicted.

B = Simulation result.

Graphs Z1.1 Z2.2 and Z3.3

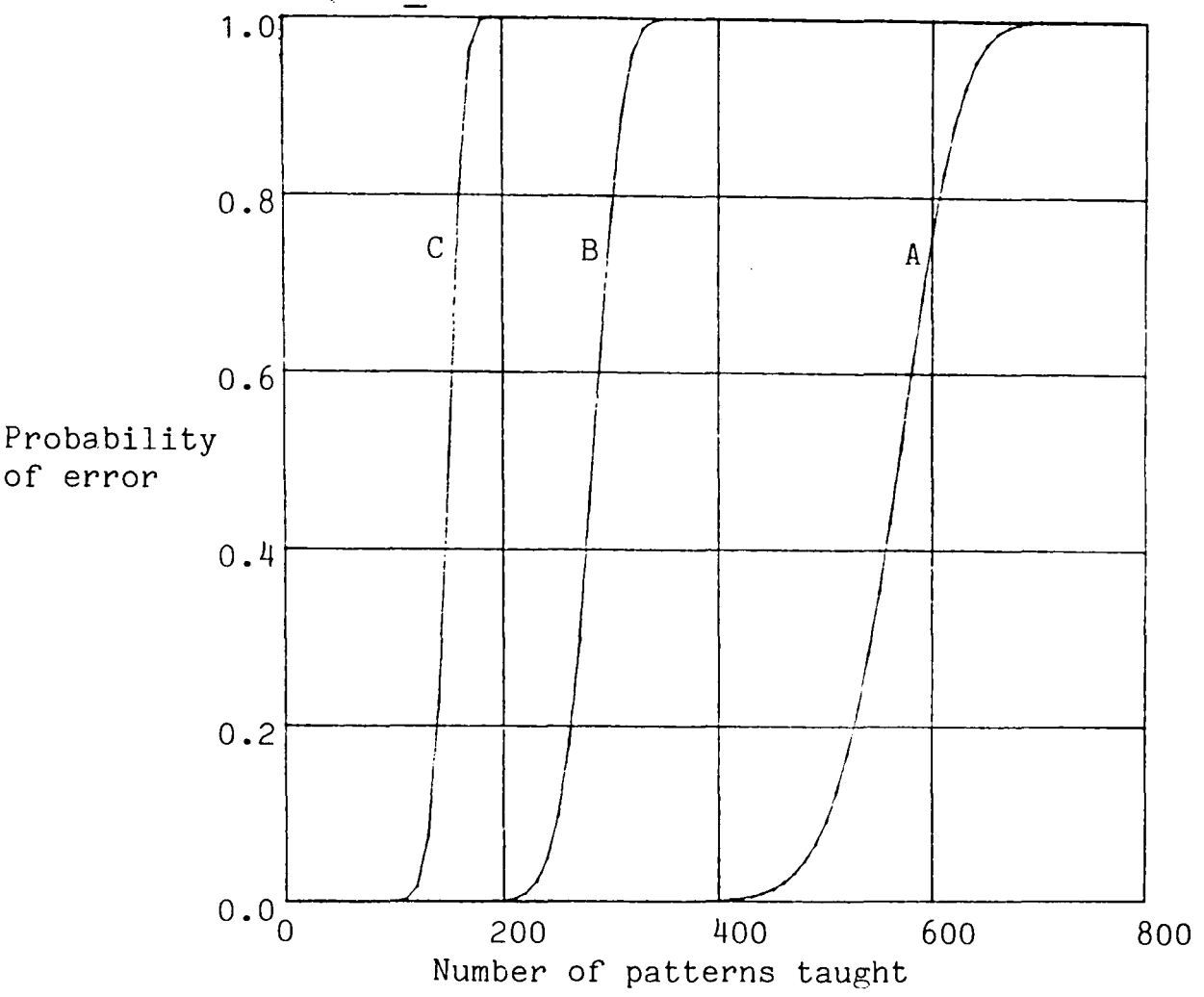
These graphs compare the mathematical and simulated predictions for the number of class points set to 1 needed to get optimal storage in the class to picture memory.

Z1.1, Z2.2 and Z3.3

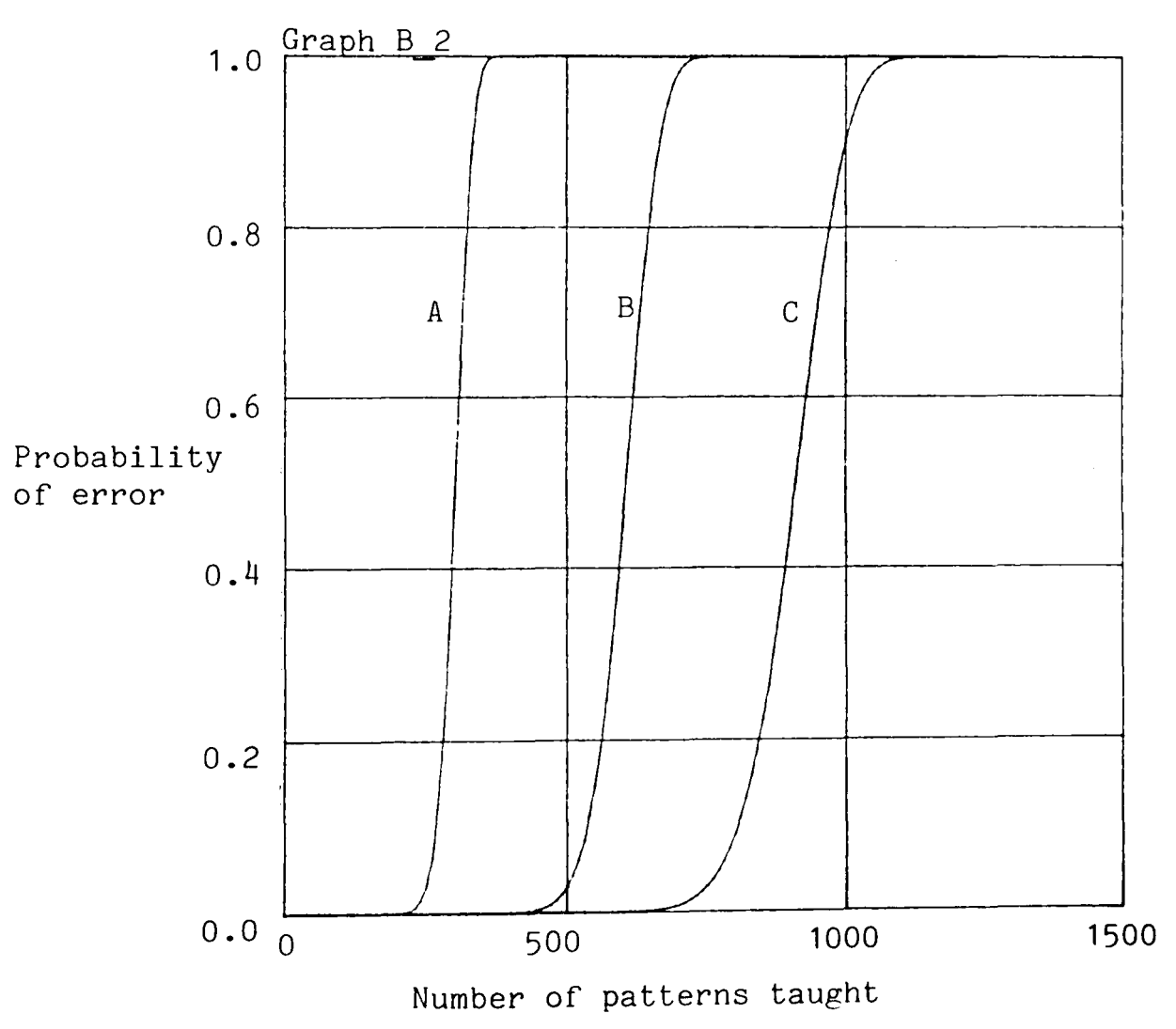
A = Mathematically predicted.

B = Simulation results.

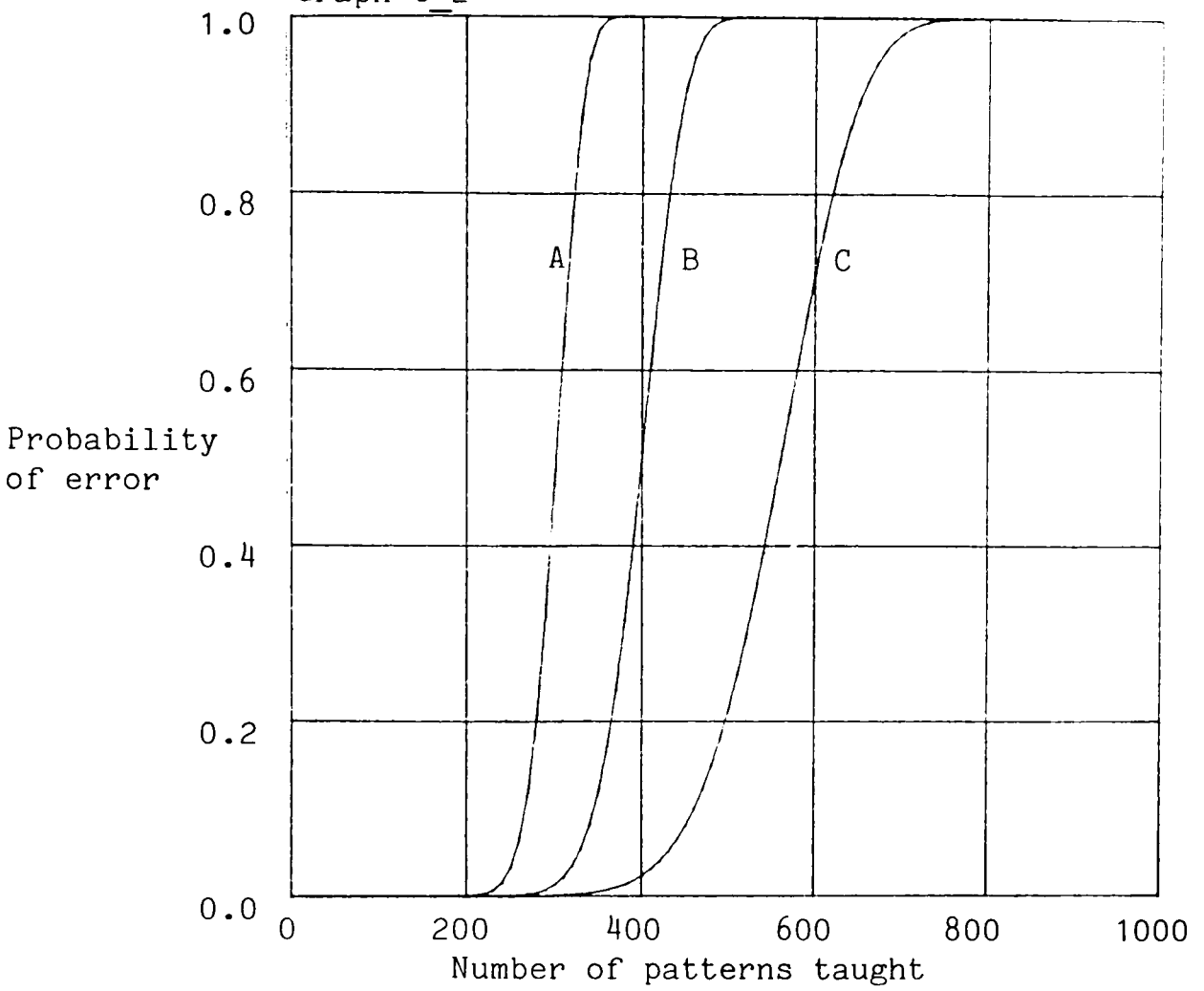
Graph A_2



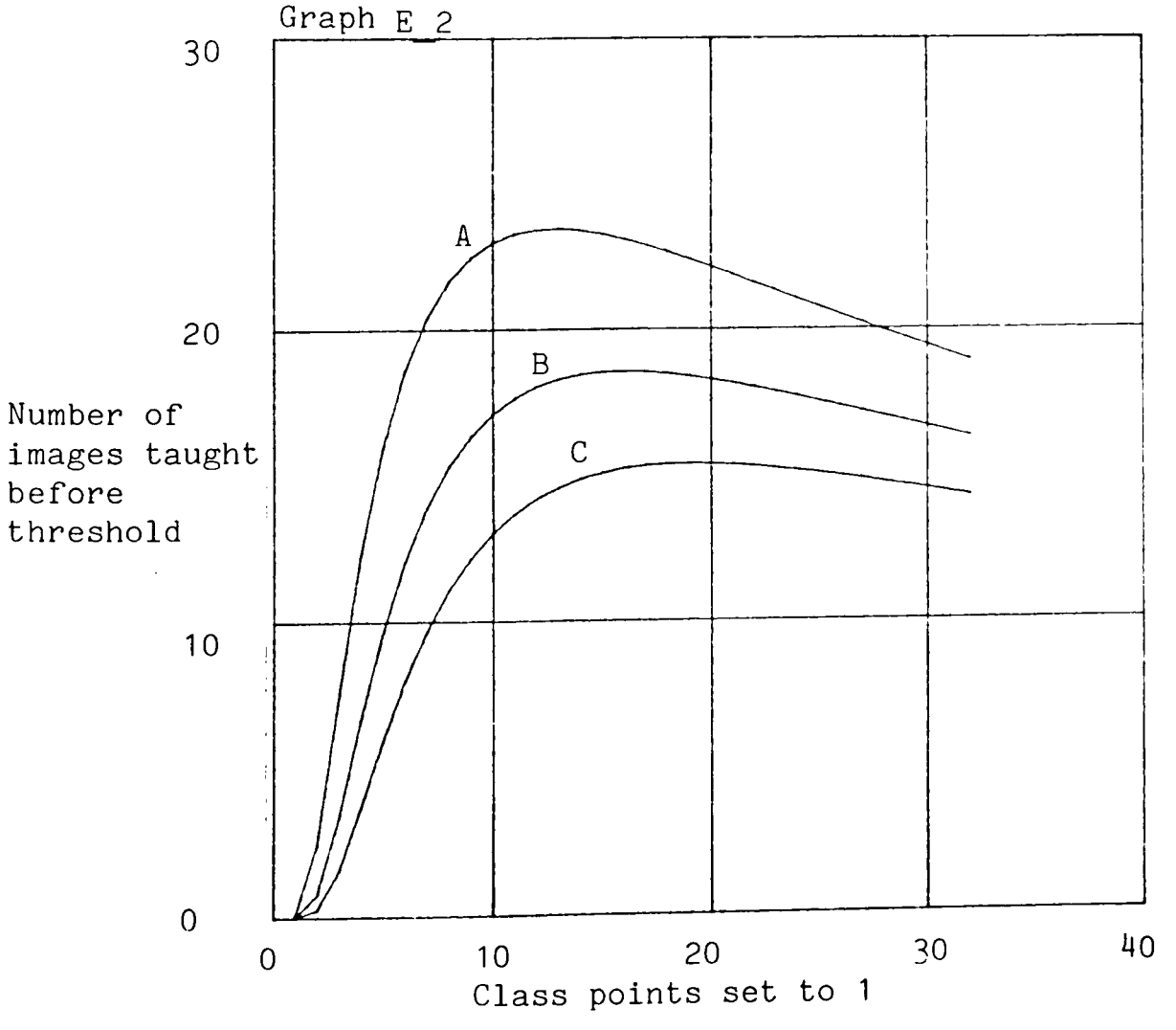
Graph B_2



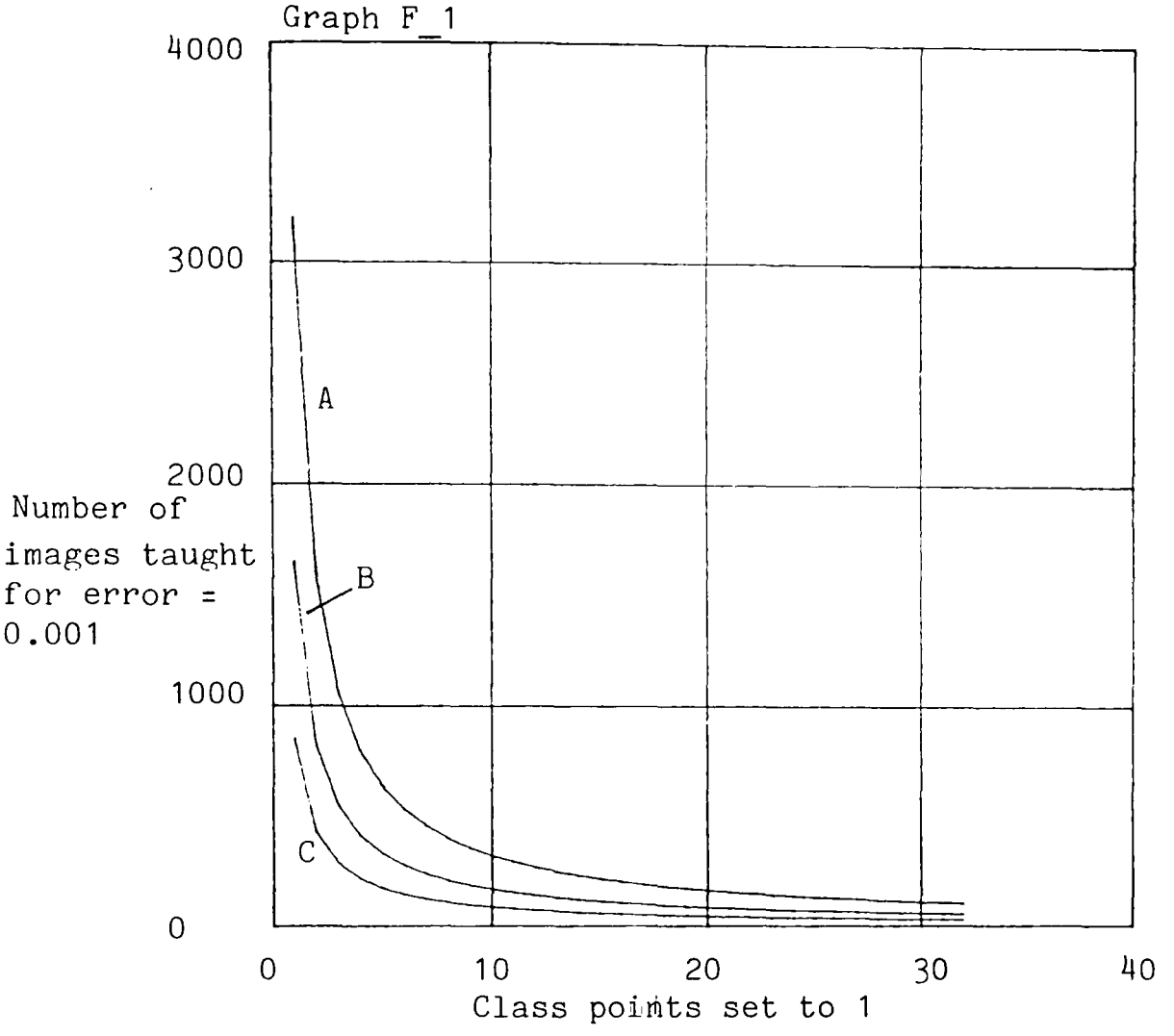
Graph C_2



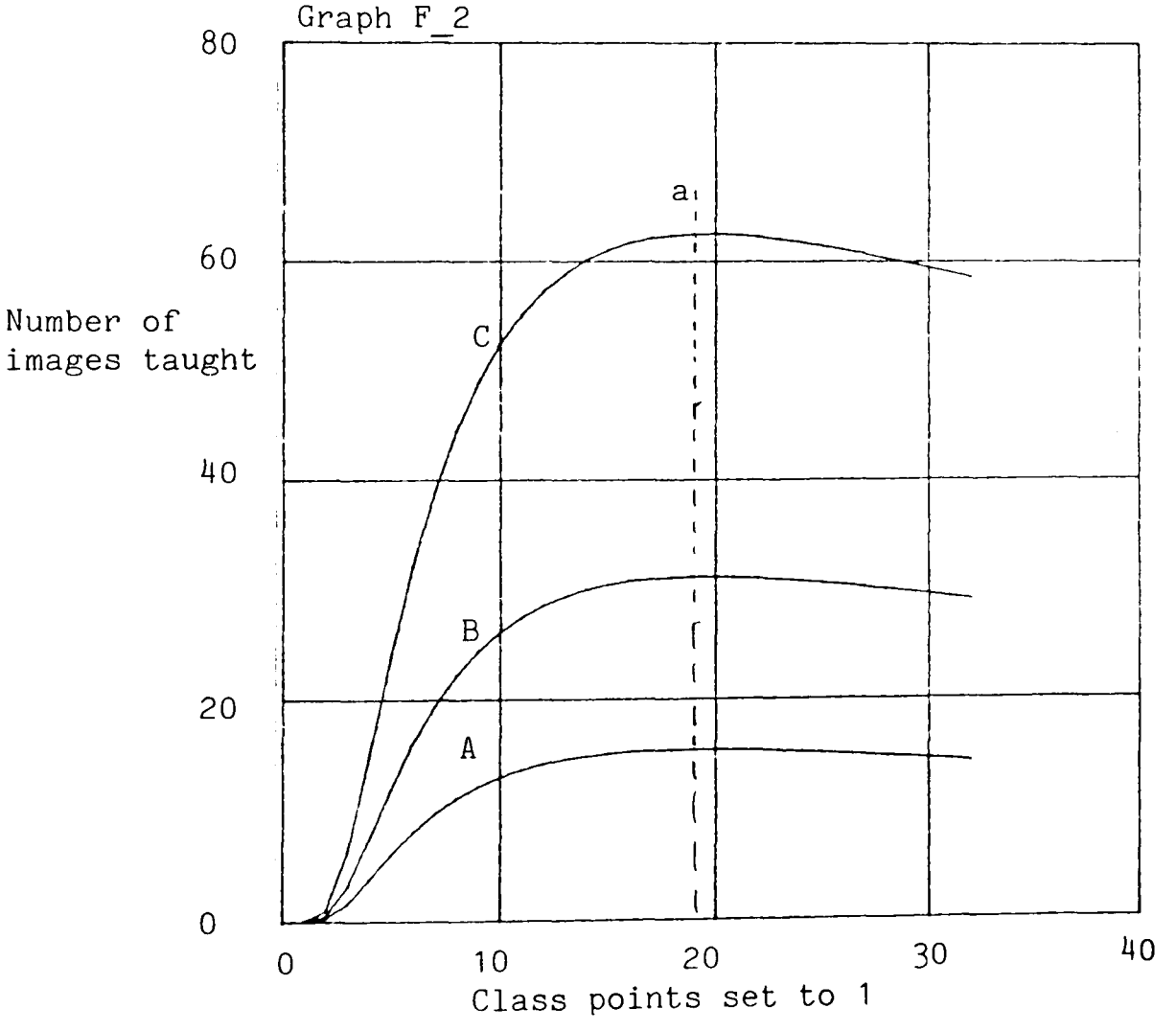
Graph E_2



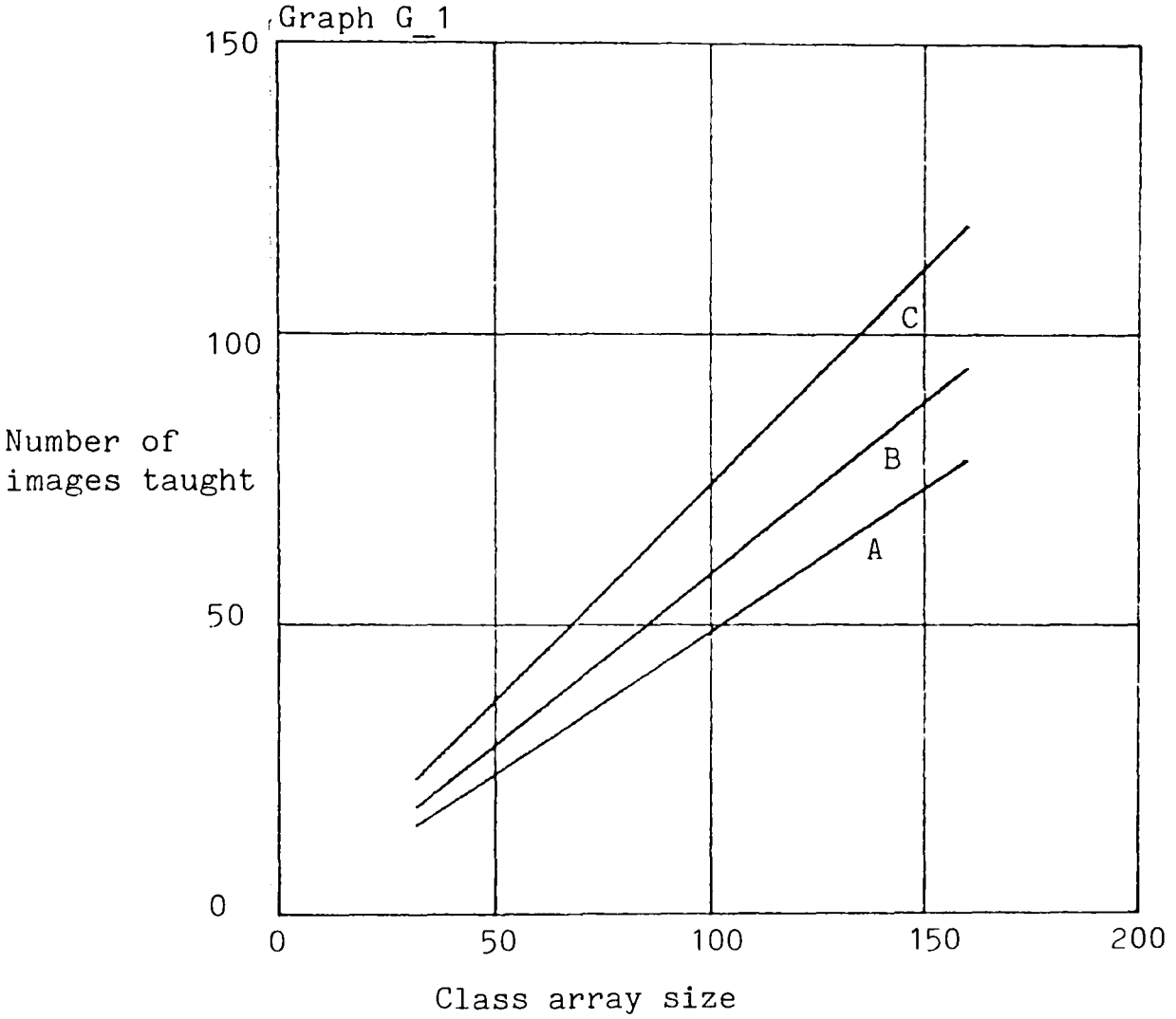
Graph F_1



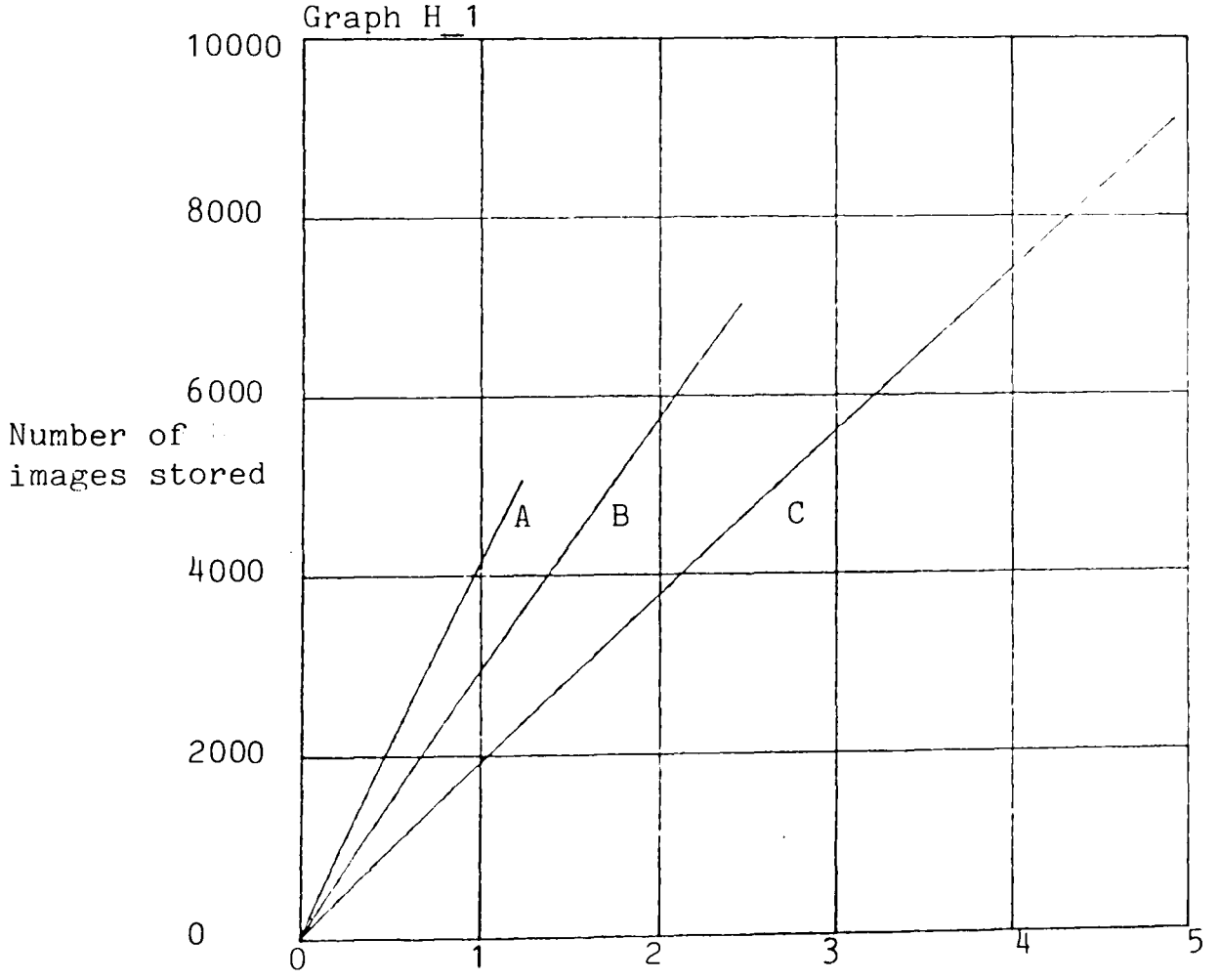
Graph F_2



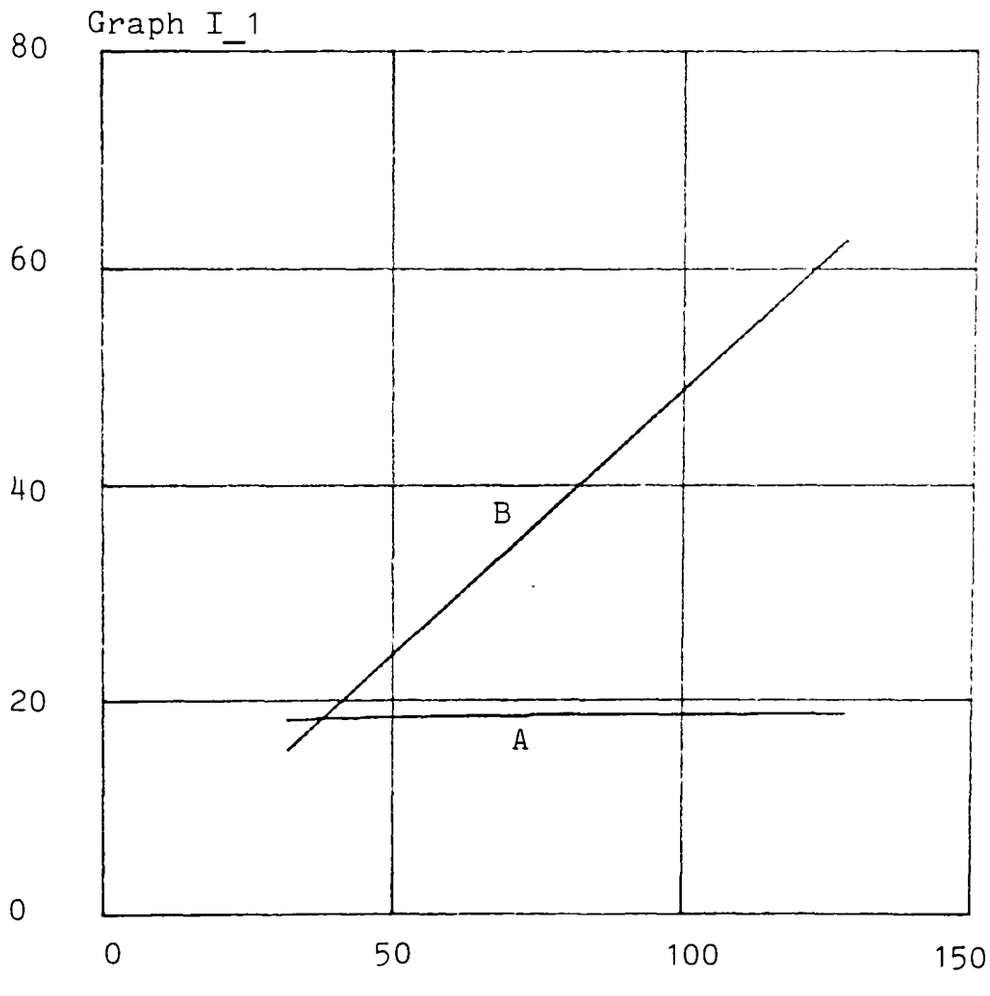
Graph G_1



Graph H_1



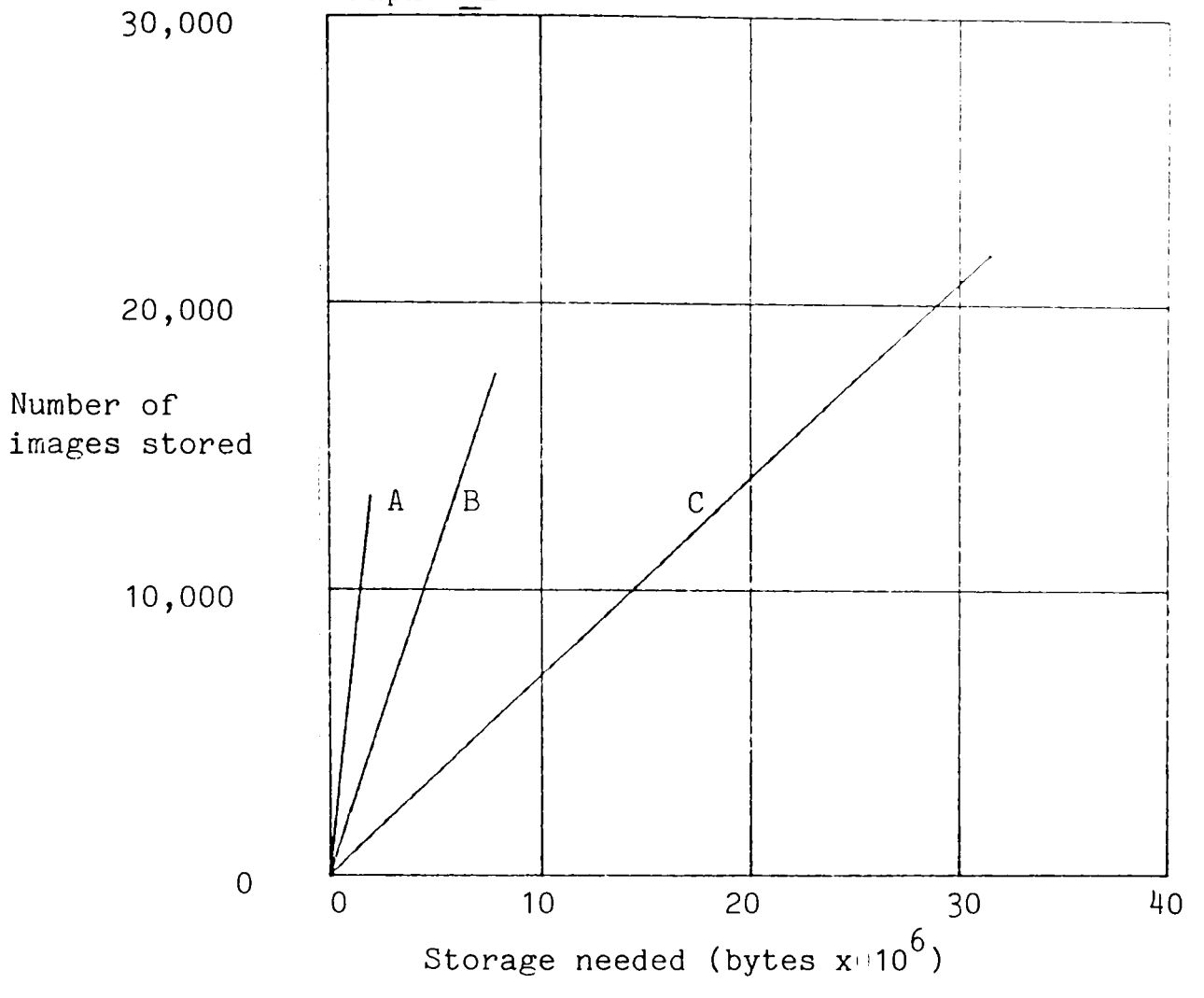
Graph I_1



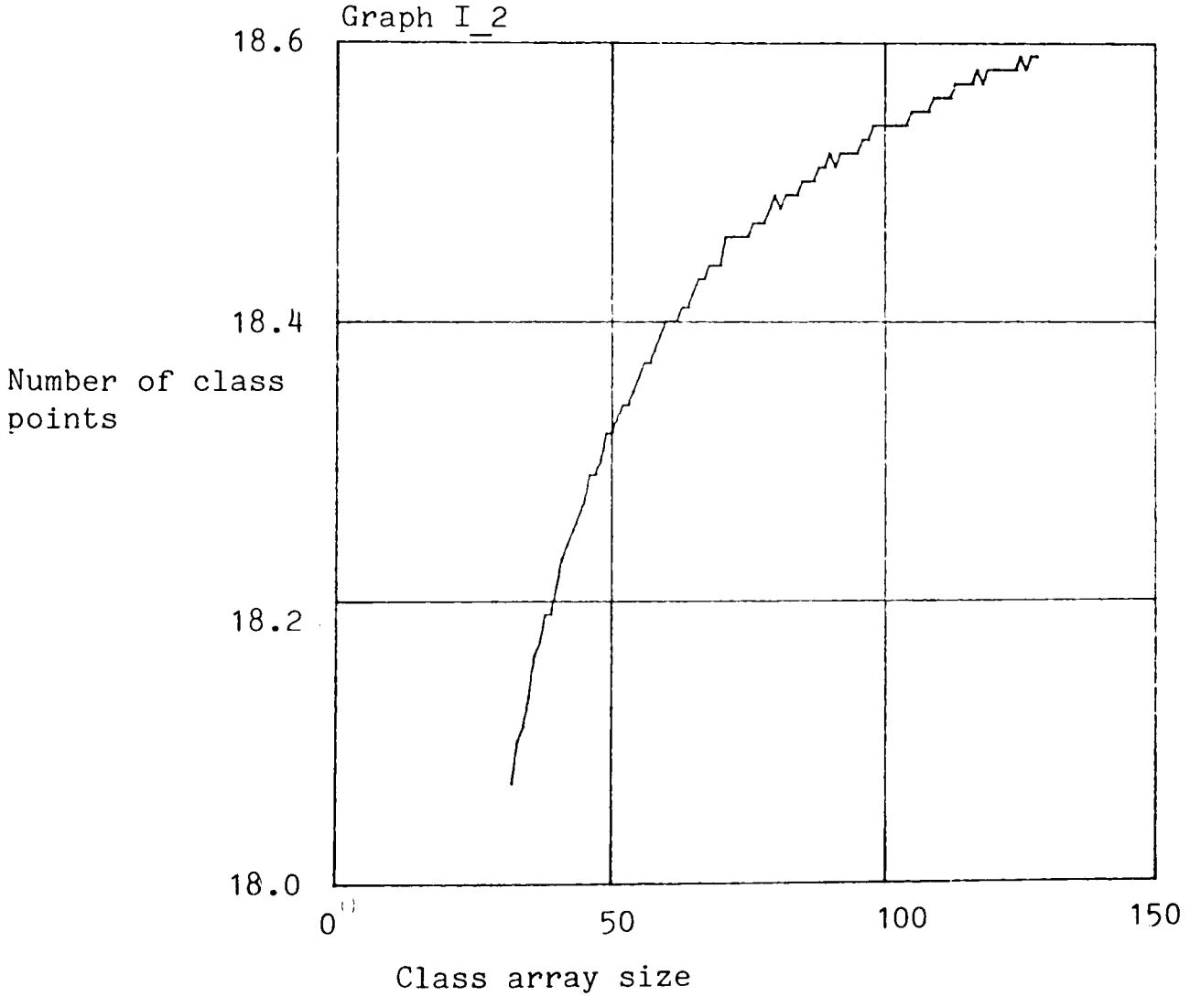
Plot A:
Number of class points set to 1.

Plot B:
Number of patterns taught.

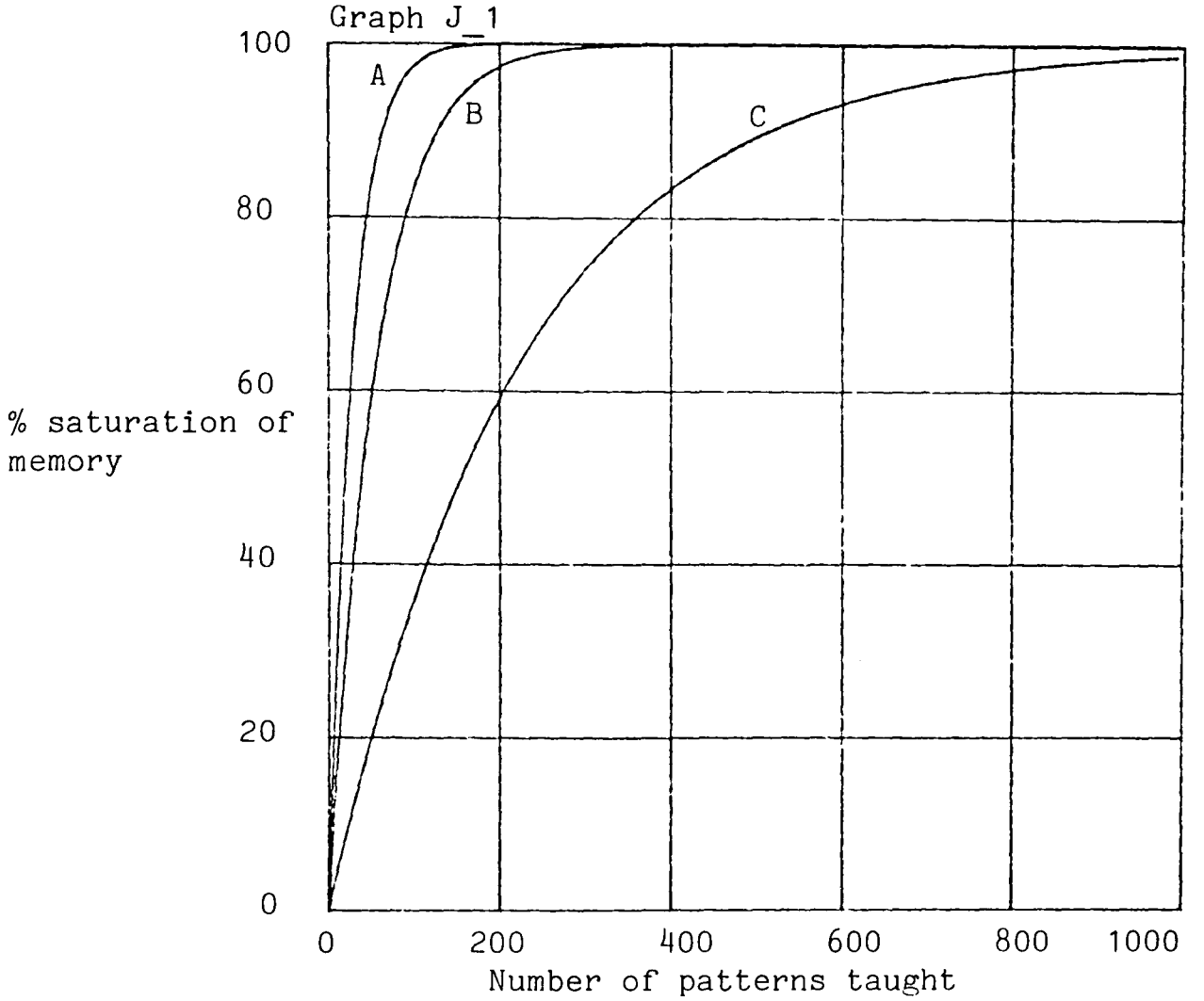
Graph H_2



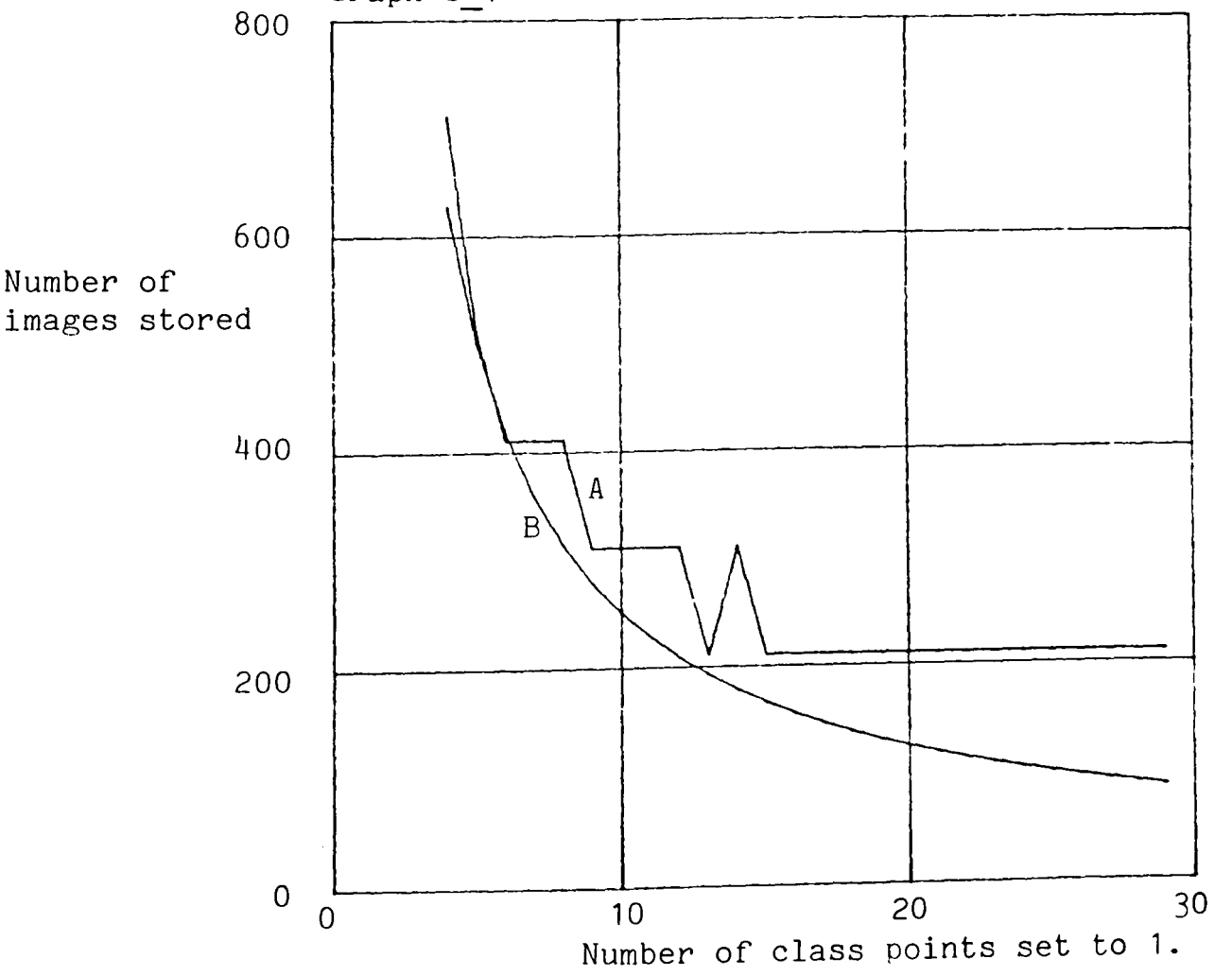
Graph I_2



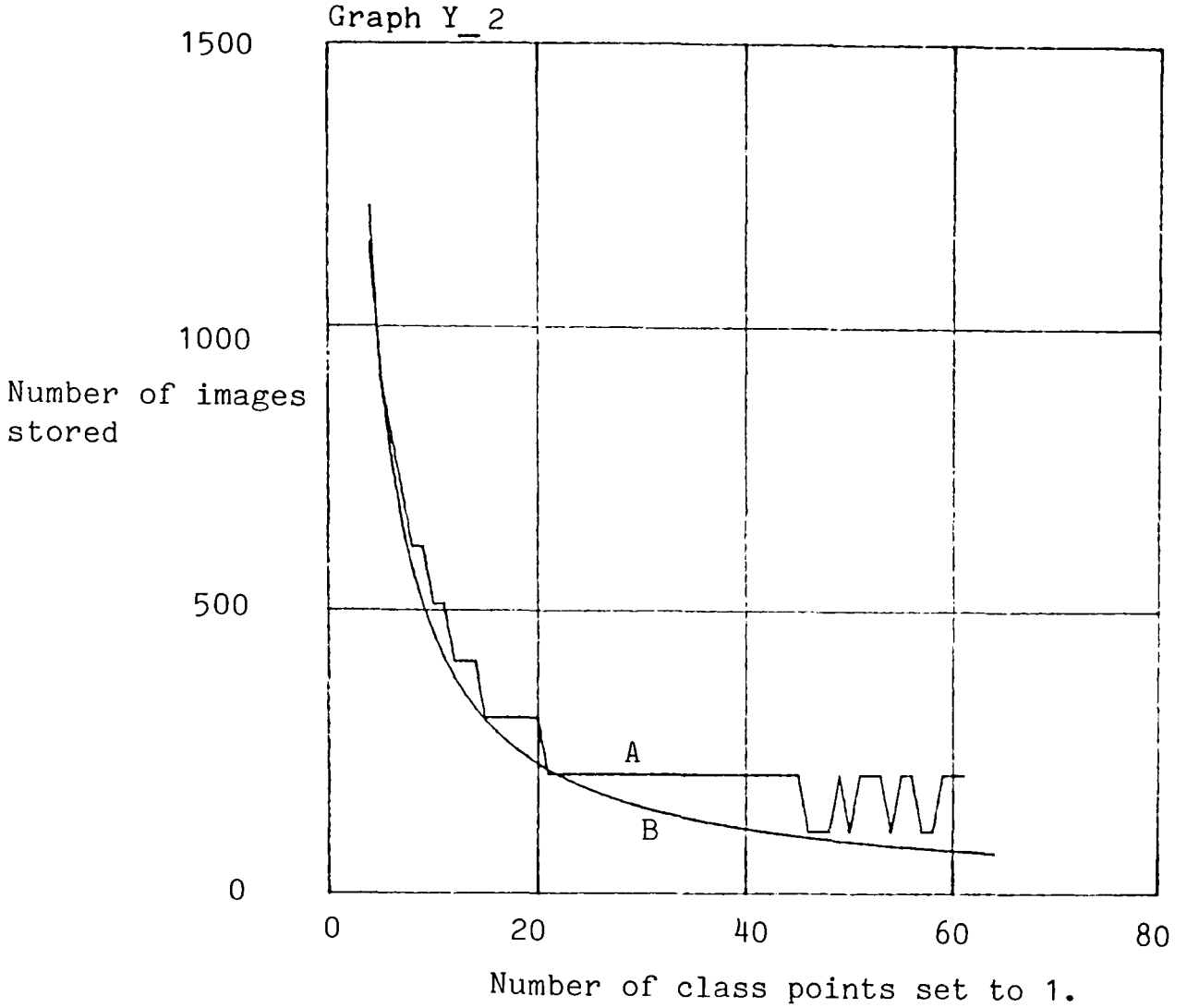
Graph J_1



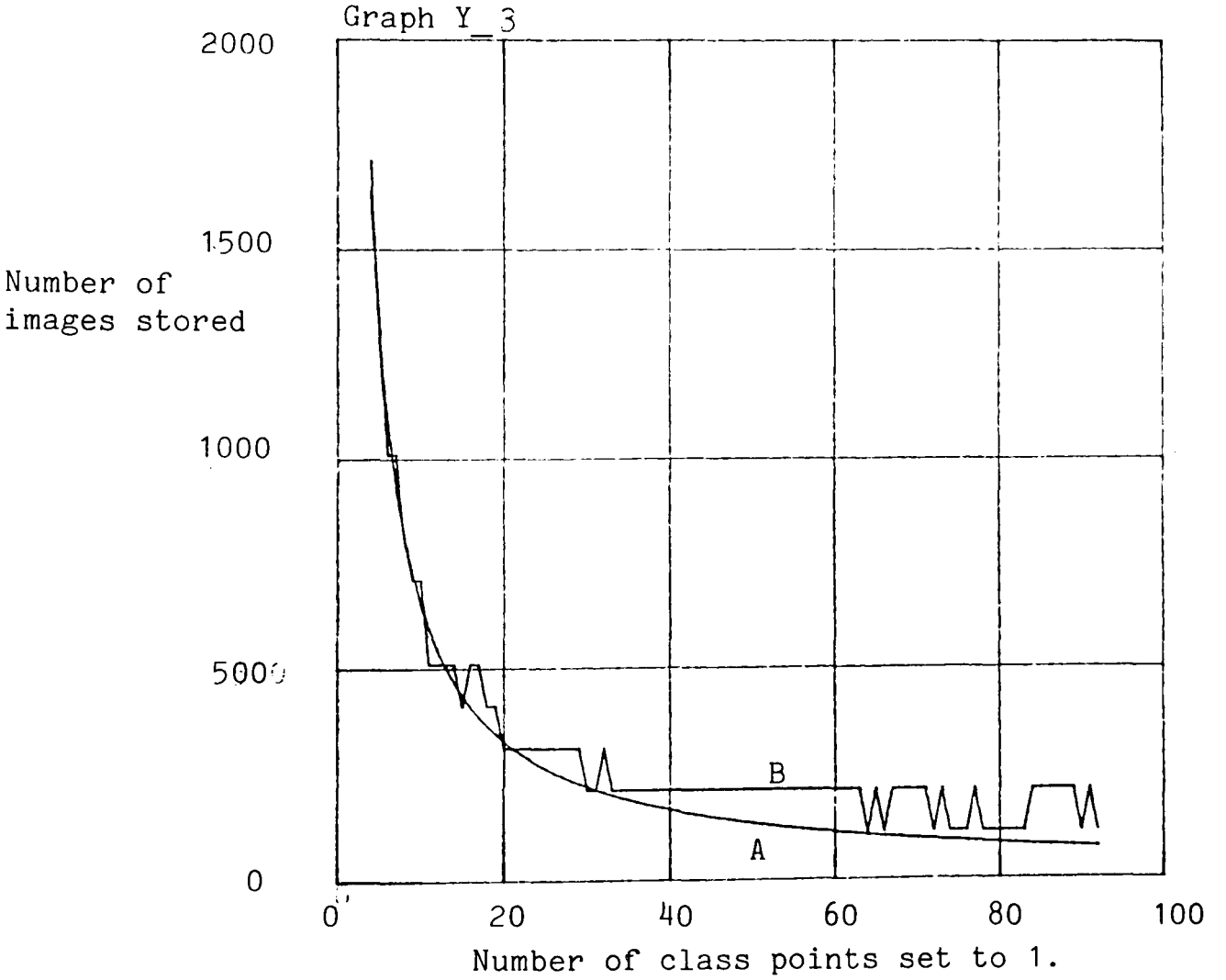
Graph Y_1



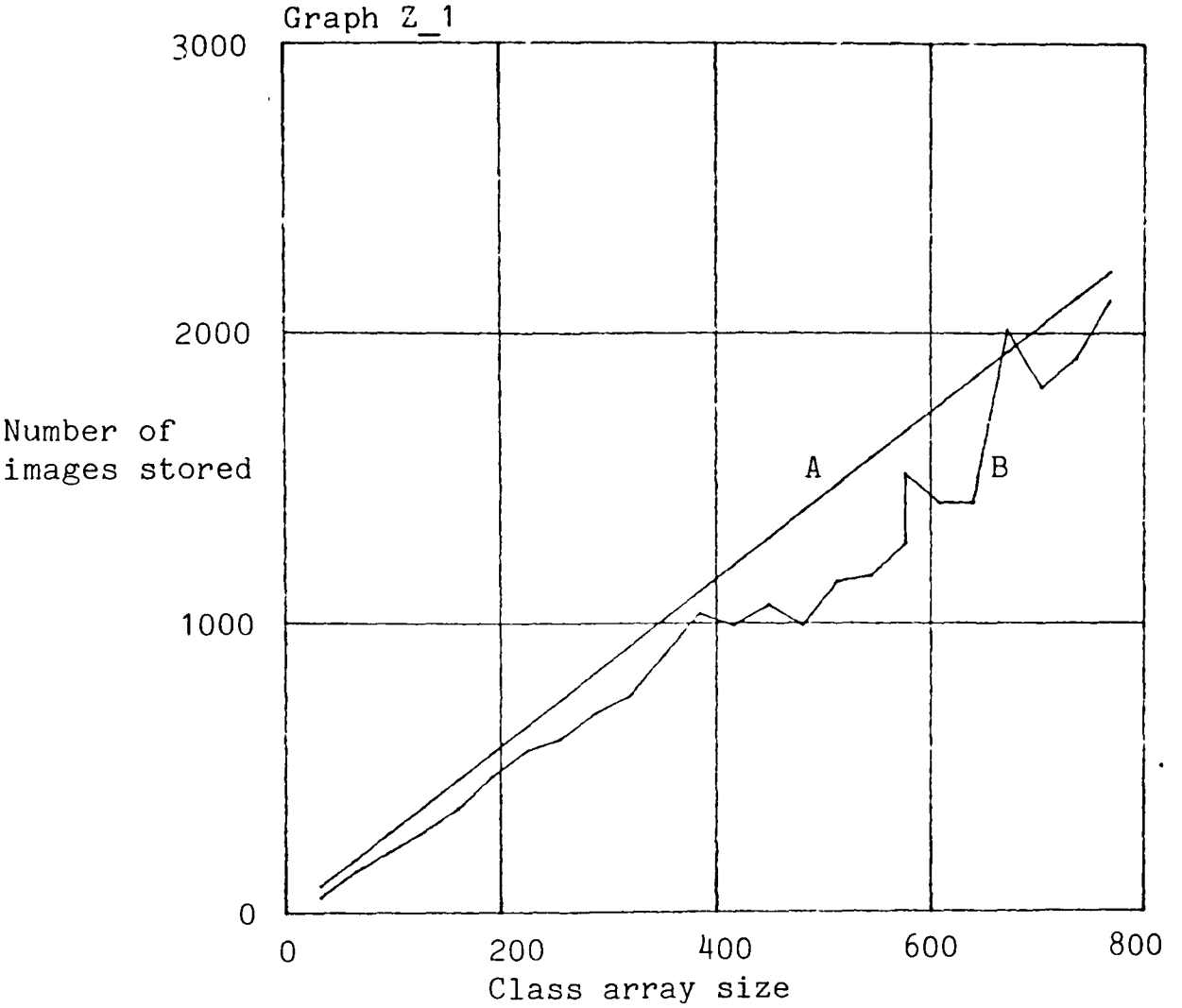
Graph Y_2



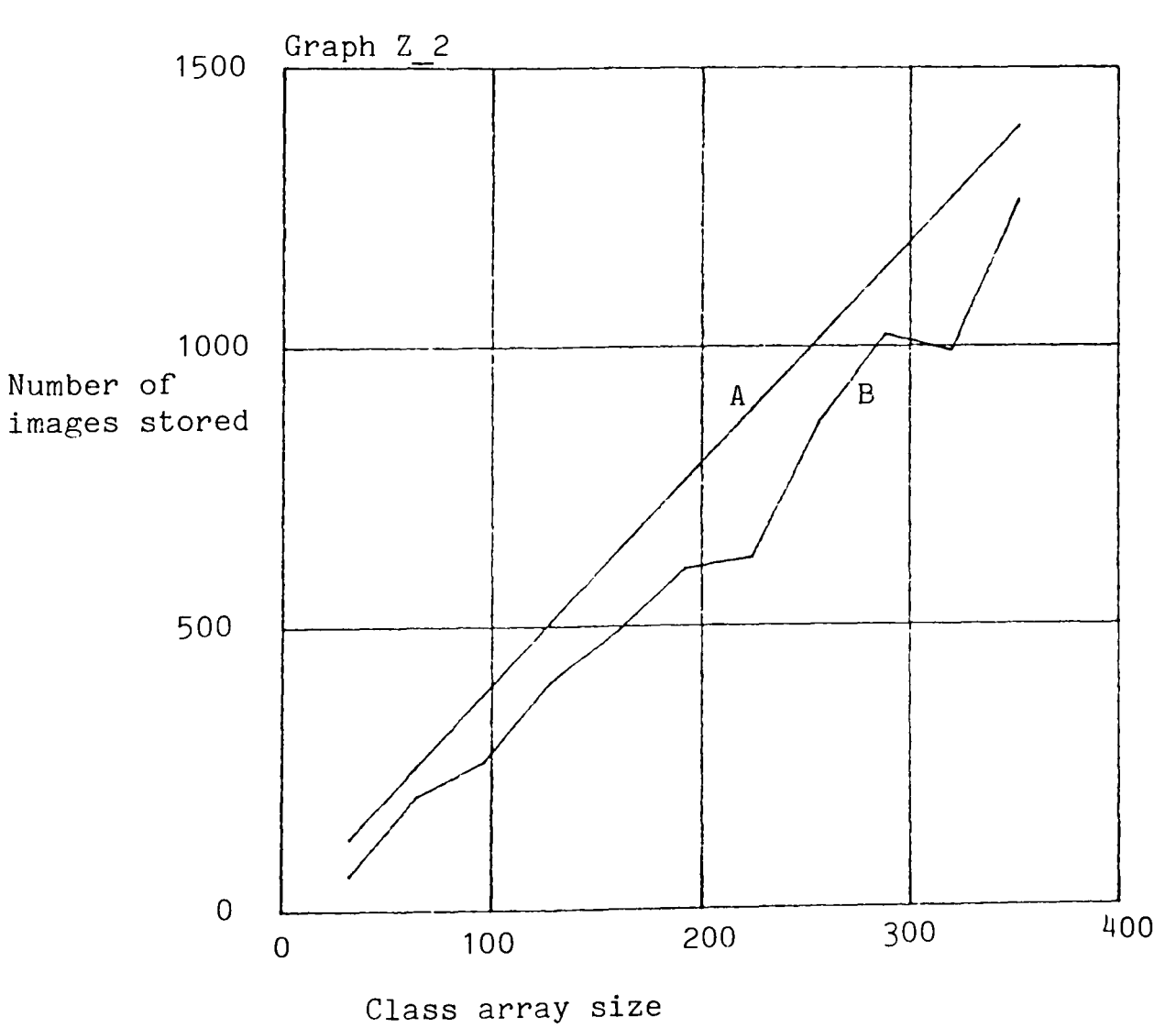
Graph Y_3



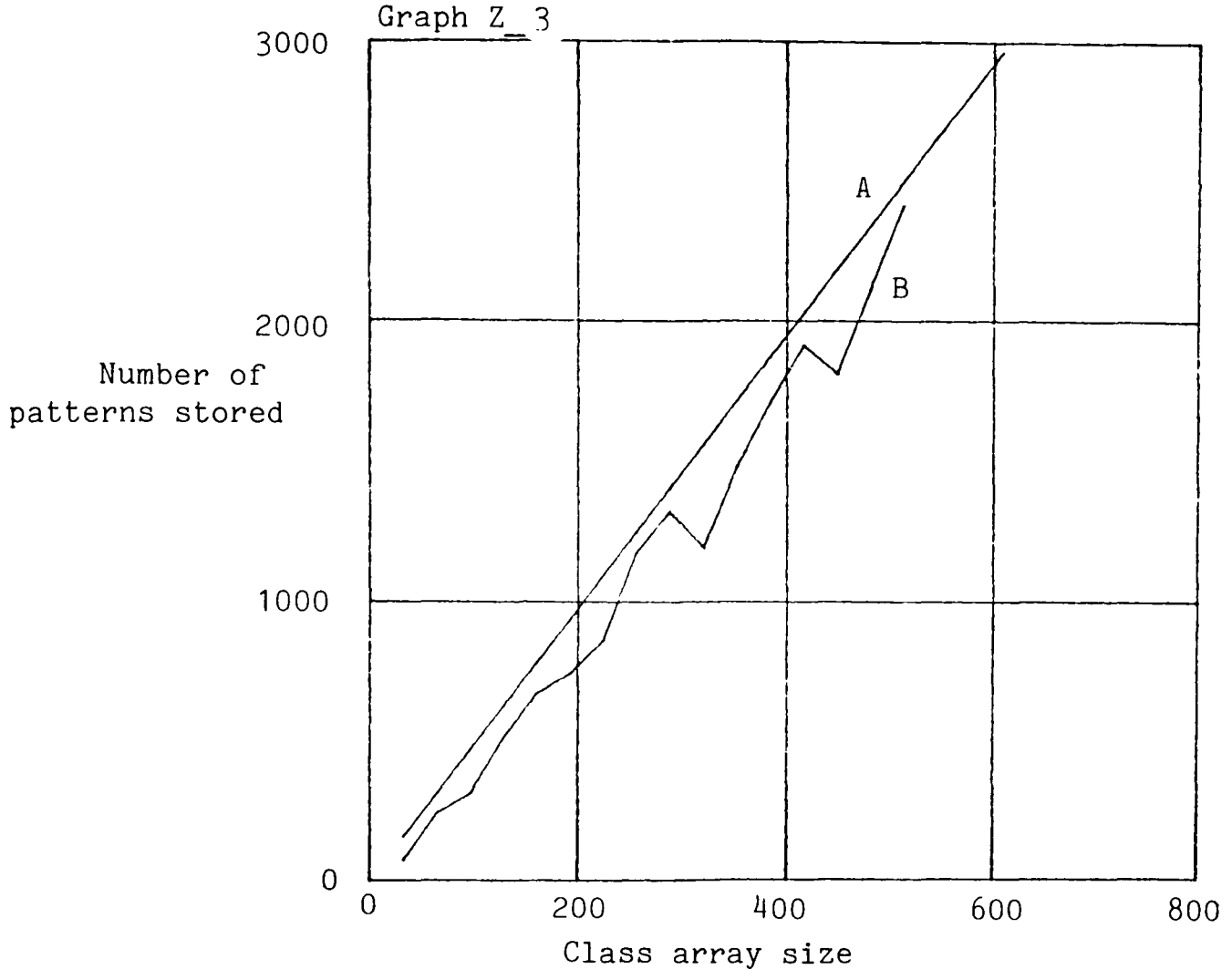
Graph Z_1



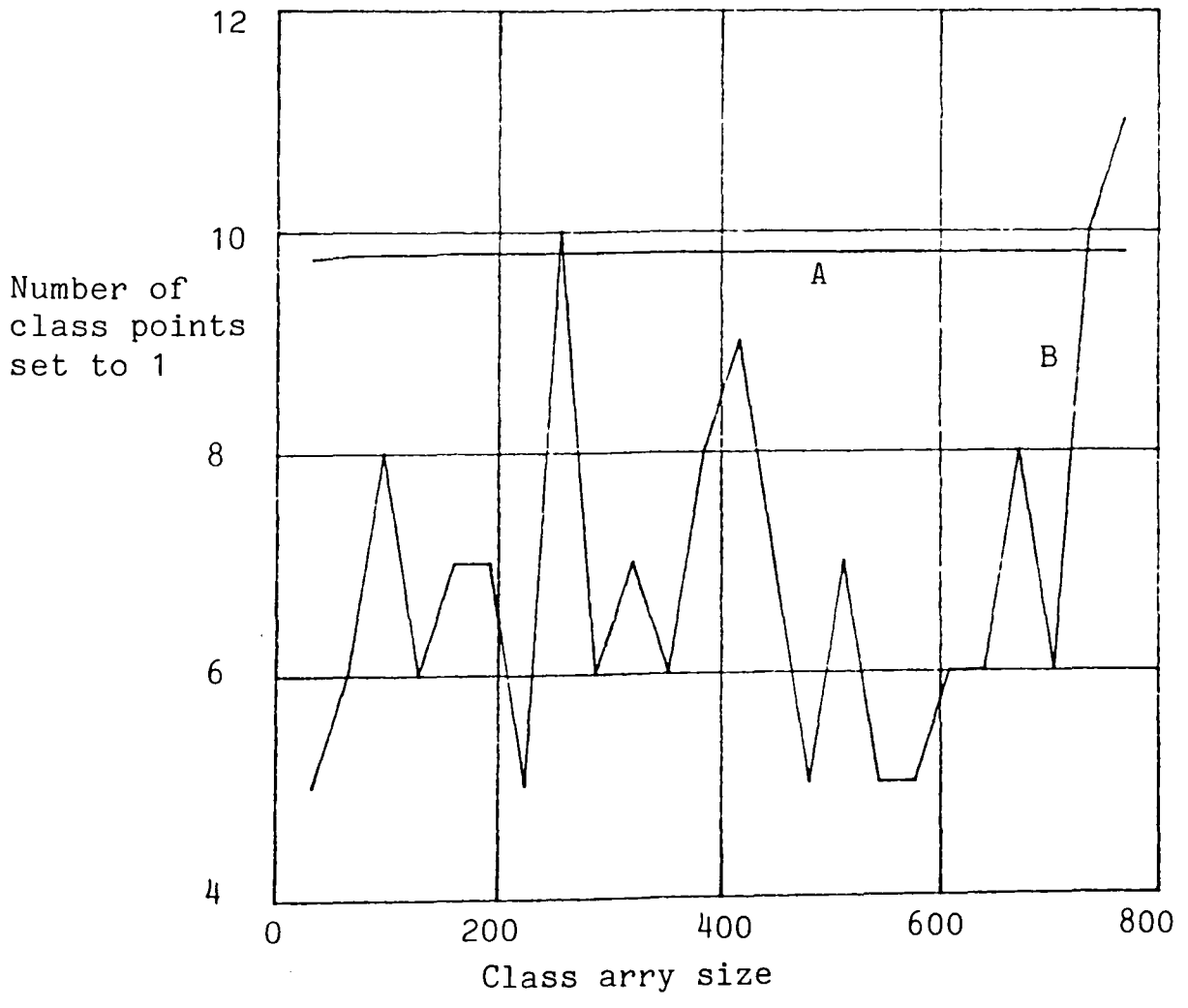
Graph Z_2



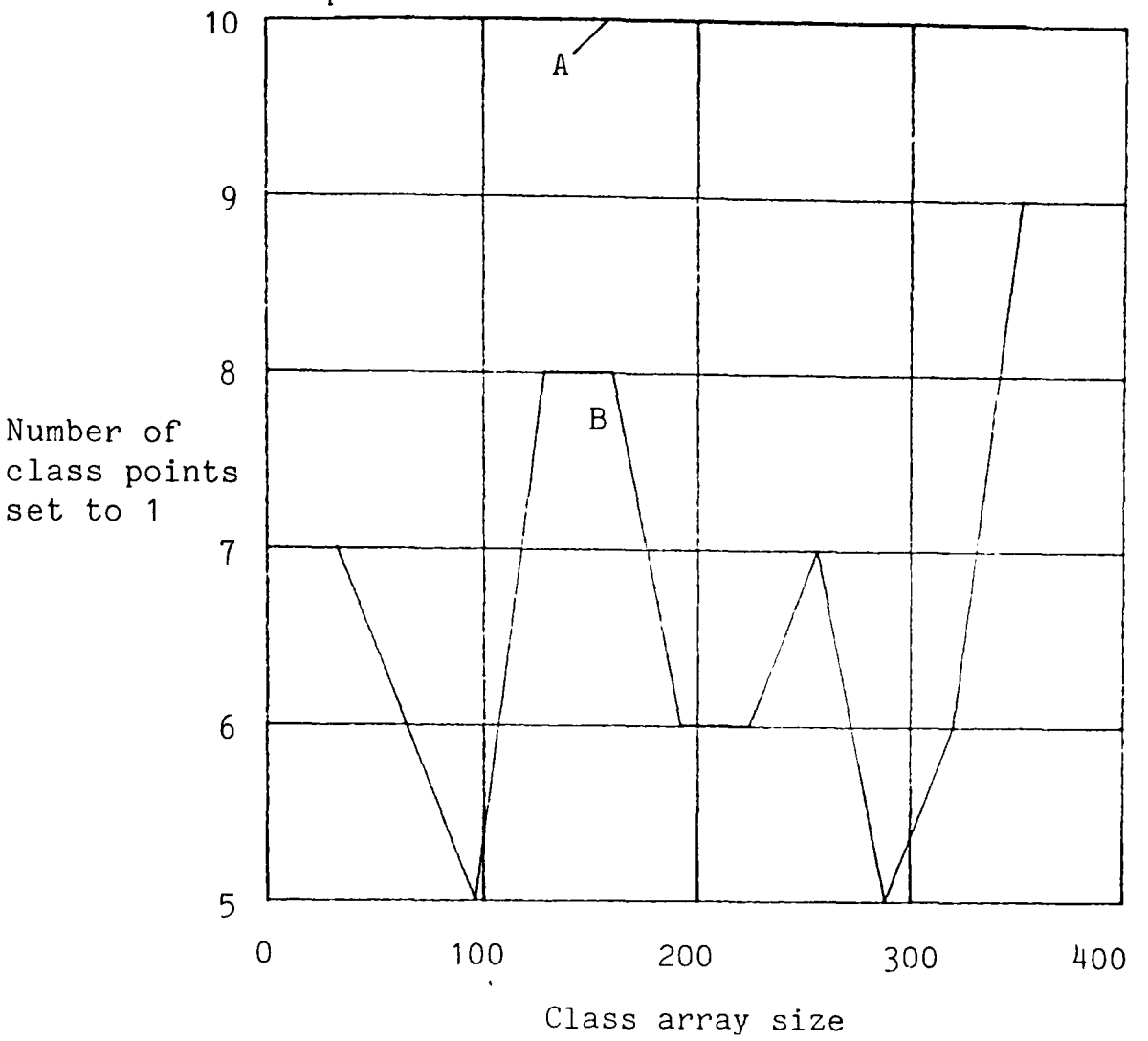
Graph Z_3



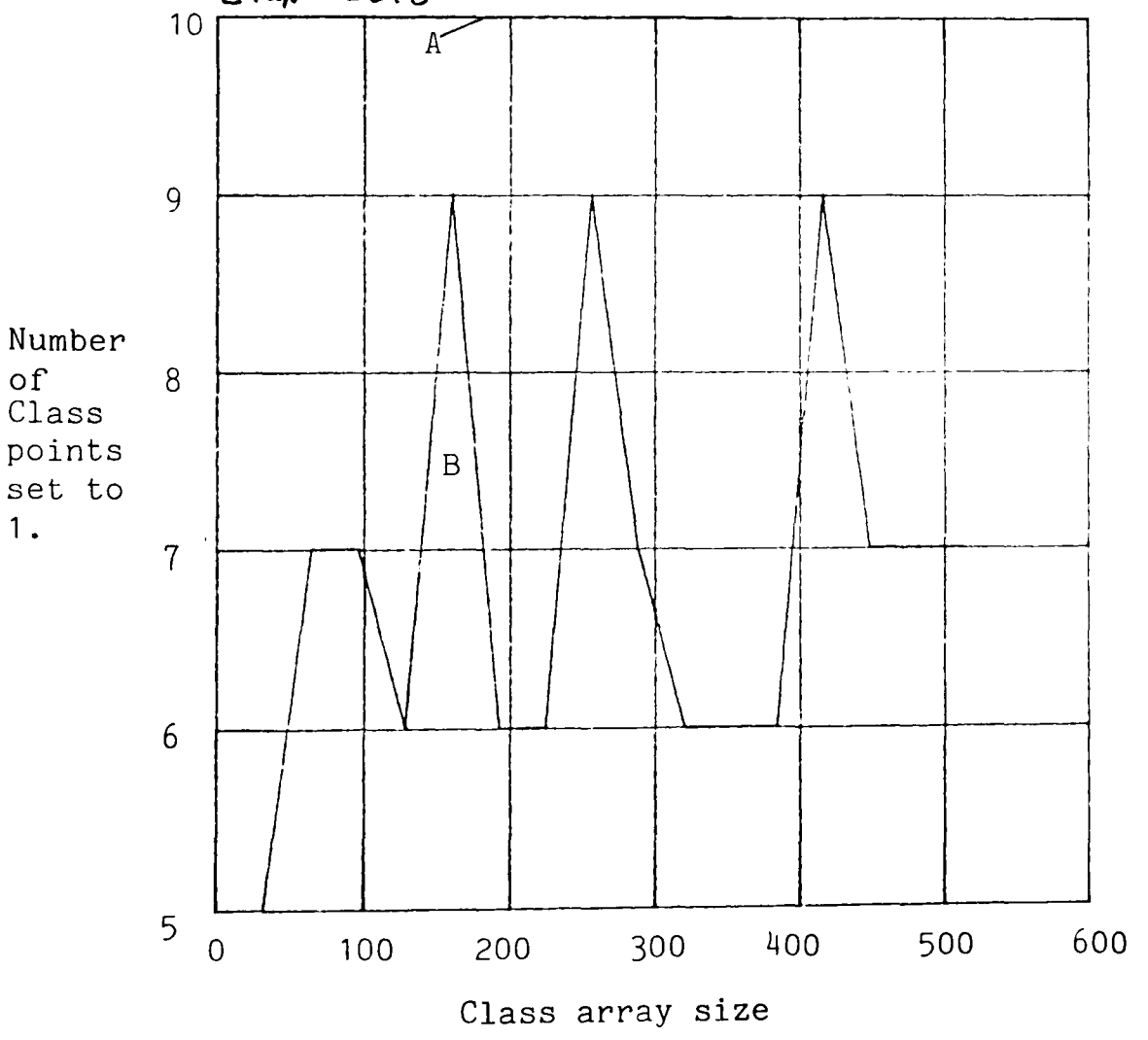
Graph Z1.1



Graph Z2.2



Graph Z3.3



6. APPENDIX 6 : Grey Level Images

The following describes the images given in the following pages, see text for details of explanation.

op1

a : An input image of 128 x 128 pixel resolution reduced to 64 x 64 pixel resolution. 64 grey levels, contrast setting 1.8.

b : N tuple codes for the image above, non thresholded, ranked process- top rank N tuple shown.

op2

Bi : N tuple codes for a grey scale representation of the square shown in op3, after 100 random points of noise have been placed in op3.

Bii : The confidence values of the N tuple codes given in Bi.

Ai : The N tuple codes for a binary representation of the image given in op3.

Aii : The confidence values of the N tuple codes given in Ai.

op3

Image used in op2.

op4

a : The N tuple codes for the image shown in op1 a, after thresholding N tuple codes below threshold away.

b : The confidences of the tuples given in `a`.

op5

a : N tuple codes and confidences of a ´star´ image (see op6).

b: As for `a` but for n image from a camera with its `f` stop reduced by 3 compared to that used in `a`.

op6

Top : image of a ´star´ 128 x 128 pixels 64 grey levels.

Small images : response of edge operators of the prefered orientation given when scanned over the star pattern : edge operator size = 16 x 16 pixels, sampled every 8 pixels.

op7

Images of squares of 128 x 128 pixels 64 grey levels, each scanned with a 90 degree edge detector of 16 x 16 pixels. Top shows a solid square with the response of the edge operator to the right, below it is the same for a line drawn square.

op8

Image of the line drawn square shown in op7, and the N tuple codes generated.

op9

The results of processing the filled square in op7 with 4 edge detectors. The response of the 4 edge detectors are summed at each sample and given as the responses shown. A grey scale image of these responses is also given.

op10

Images using edge operator primitives to find areas of interest in

various scenes.

Images processed:

Top : Solid square shown in op7

Middle : Star pattern shown in op6

Bottom : Face shown in op1.

all 128 x 128 pixels.

op11

As for op10, but after a small shift (micro saccade) of the square.

Top : shift -2,-2 pixels

Middle : shift +2.+2 pixels

Bottom : shift -2,+2 pixels

op12

Triangle on top of a square 128 x 128 pixels grey level.

op13

Images of a square and a treiangle 128 x 128 pixels grey level.

Fig a

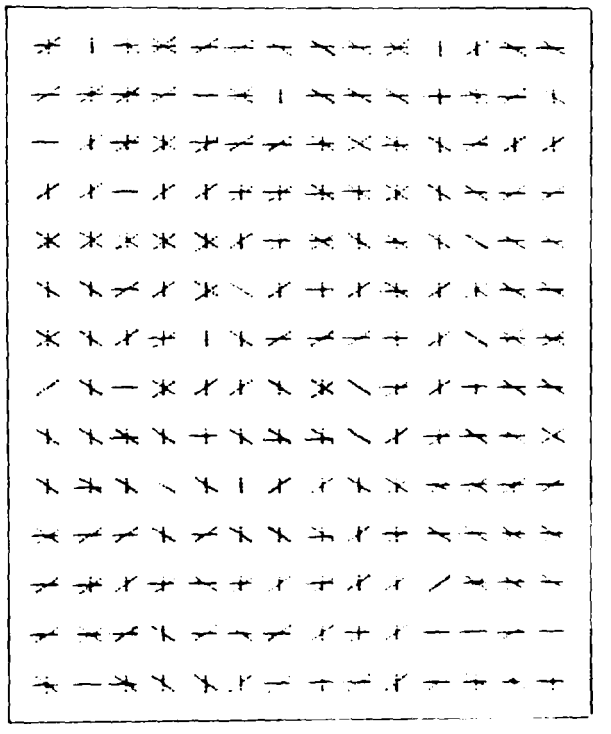


Fig b

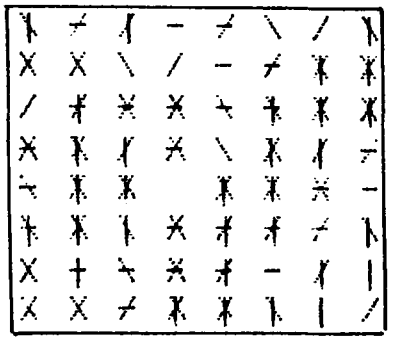


Fig b_i

73	52	74	16	72	48	12	82
67	47	22	21	16	59	39	43
28	187	550	762	887	258	54	16
34	507	761	312	308	566	30	52
52	468	641	0	441	754	35	28
63	453	992	300	800	289	66	47
59	219	754	796	355	42	76	56
65	47	52	35	47	55	26	16

Fig b_{ii}

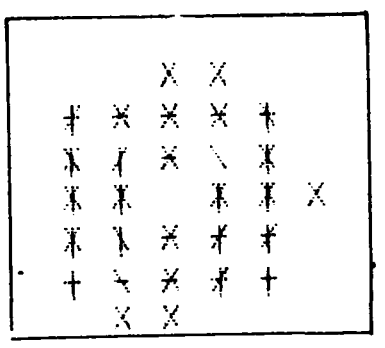
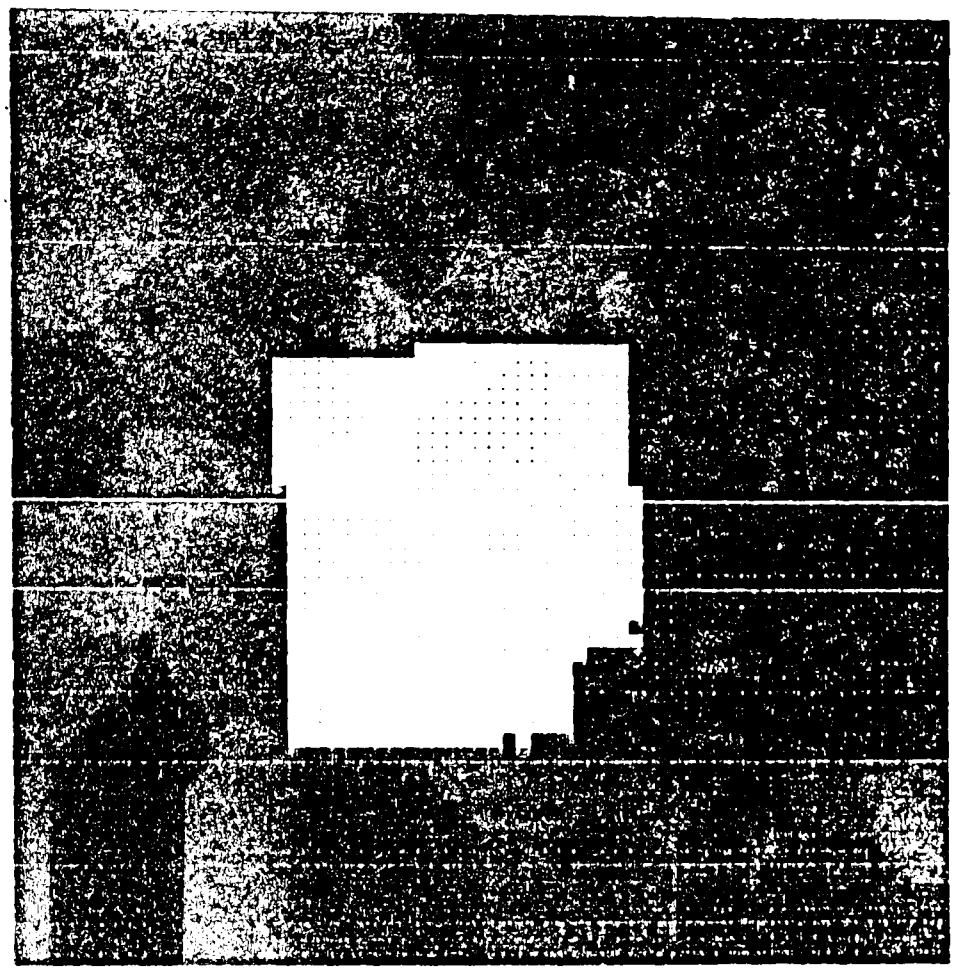


Fig a_i

0	0	0	0	0	0	0	0
0	0	0	46	46	0	0	0
0	187	516	781	632	250	0	0
0	547	856	336	344	601	0	0
0	507	686	0	441	785	14	0
0	453	977	347	831	328	0	0
0	218	718	792	328	78	0	0
0	0	14	14	0	0	0	0

Fig a_{ii}



op4

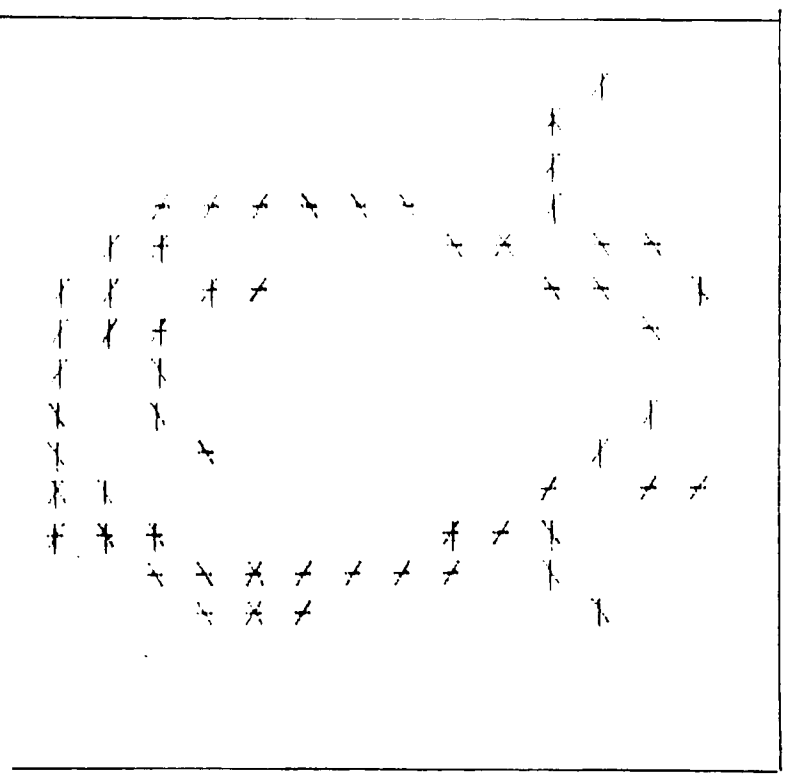
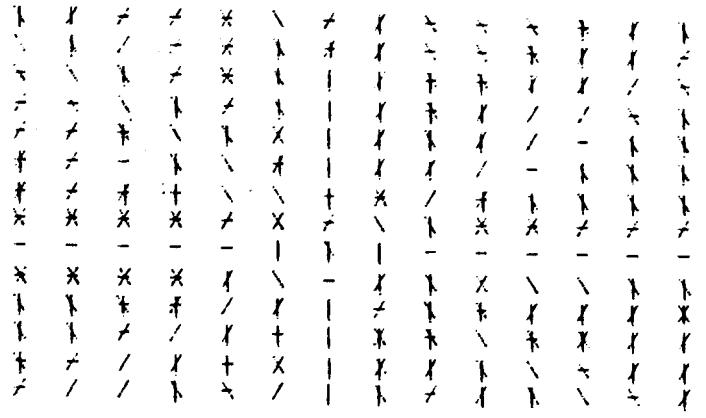


Fig a

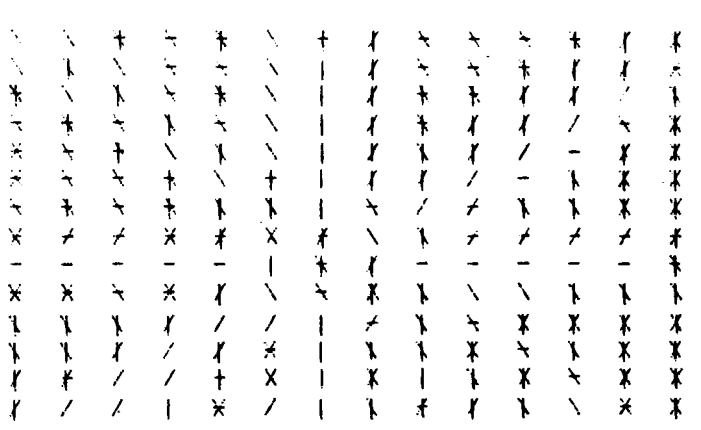
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	327	0	0	0
0	0	0	0	0	0	0	0	0	0	0	190	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	382	0	0	0	0
0	0	0	496	805	760	538	437	360	0	0	345	0	0	0	0
0	0	605	421	0	0	0	0	0	361	208	0	443	372	0	0
0	233	798	0	233	185	0	0	0	0	0	304	386	0	300	0
0	744	265	204	0	0	0	0	0	0	0	0	0	312	0	0
0	847	0	343	0	0	0	0	0	0	0	0	0	0	0	0
0	741	0	252	0	0	0	0	0	0	0	0	0	308	0	0
0	710	0	0	193	0	0	0	0	0	0	0	294	0	0	0
0	194	628	0	0	0	0	0	0	0	0	188	0	336	276	0
0	214	560	359	0	0	0	0	0	332	472	423	0	0	0	0
0	0	0	746	736	379	638	698	802	383	0	472	0	0	0	0
0	0	0	0	288	410	305	0	0	0	0	0	206	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig b



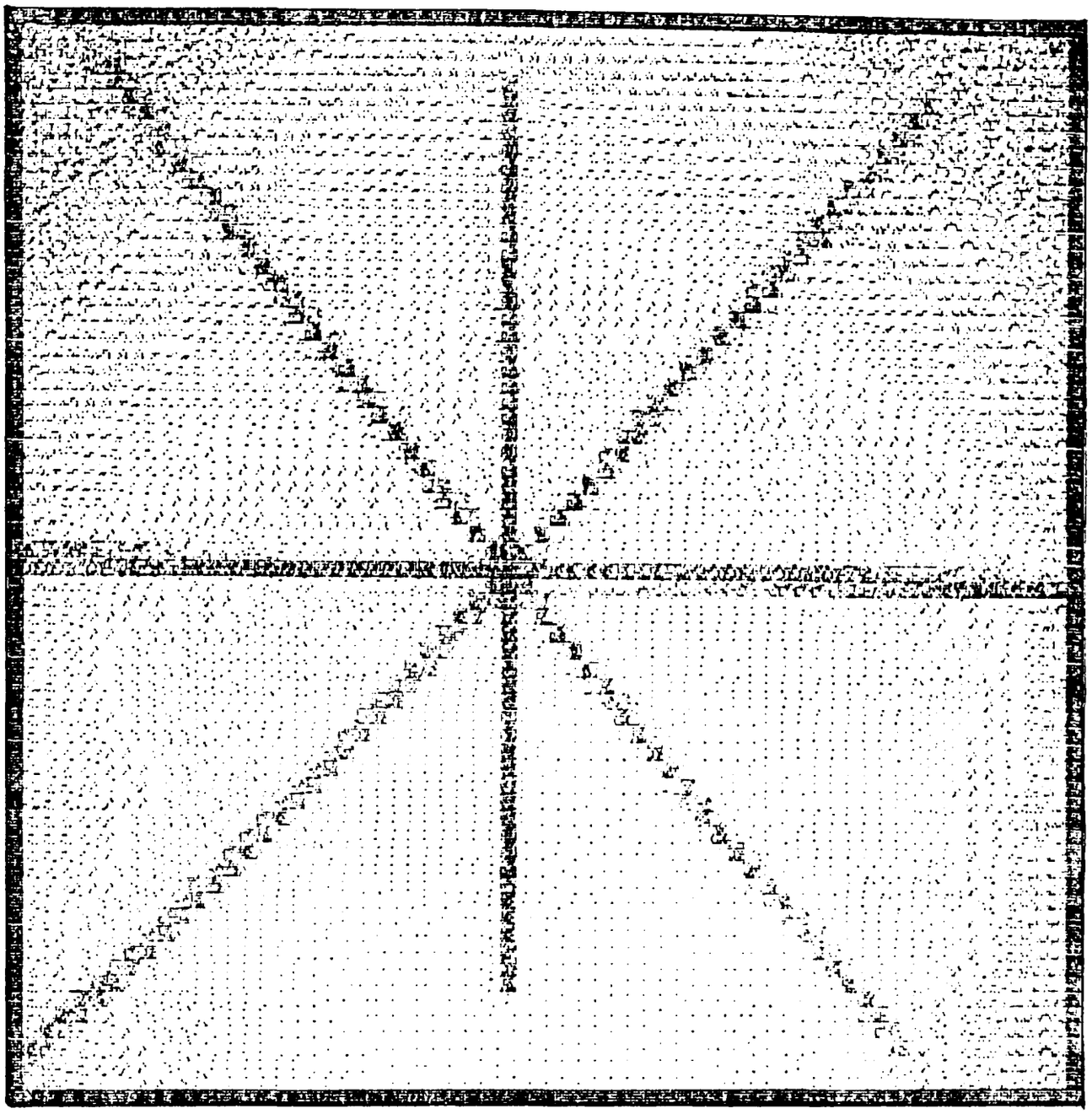
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	185	69	63	65	55	53	52	128	71	66	67	51	201	85	0
0	94	207	15	57	39	128	30	128	52	60	47	188	134	181	0
0	212	51	197	42	26	129	27	151	32	35	152	224	87	87	0
0	53	219	27	191	36	121	40	163	36	97	125	103	92	69	0
0	50	52	170	133	150	107	63	174	81	260	120	56	70	69	0
0	30	37	29	238	144	45	90	153	282	136	55	50	72	69	0
0	21	18	14	84	135	64	163	102	146	143	44	57	68	84	0
0	176	177	171	174	148	190	269	105	169	181	185	253	261	254	0
0	50	67	70	75	83	86	179	144	112	143	149	151	144	132	0
0	79	79	89	84	162	67	78	246	174	98	43	45	113	137	0
0	25	30	11	168	111	210	129	135	60	179	35	53	71	63	0
0	27	25	209	148	261	38	161	189	194	32	155	44	69	81	0
0	39	205	155	244	79	50	168	207	28	268	111	195	75	50	0
0	205	148	114	86	35	27	162	259	52	63	252	135	168	65	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

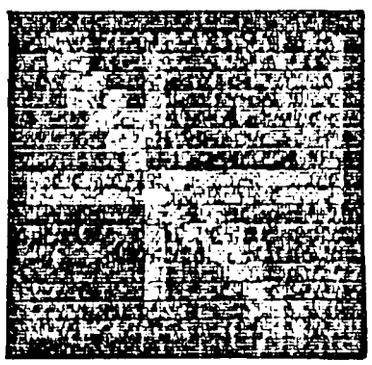
Fig a



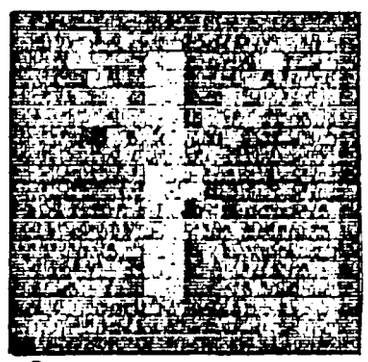
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	28	10	13	19	18	20	29	47	25	18	16	13	67	36	0
0	36	47	10	19	21	15	13	43	17	19	15	55	59	34	0
0	48	31	47	17	12	16	19	44	12	10	40	70	22	26	0
0	12	55	13	43	19	13	25	47	13	26	68	31	26	36	0
0	12	17	59	24	39	18	37	57	35	69	31	15	21	33	0
0	15	21	21	64	32	20	44	66	75	35	20	23	20	34	0
0	21	17	15	23	80	33	37	23	31	55	22	27	23	38	0
0	54	61	56	37	26	55	50	34	38	64	70	72	75	65	0
0	24	23	21	25	24	28	46	94	31	38	39	41	42	56	0
0	37	27	35	27	50	16	68	40	39	19	18	39	48	70	0
0	13	9	11	66	43	23	41	34	39	55	15	23	27	49	0
0	10	14	69	49	77	12	49	68	74	13	54	30	24	49	0
0	15	54	51	38	27	18	52	56	4	82	14	53	30	50	0
0	74	56	37	6	9	10	48	77	12	23	85	23	35	45	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig b





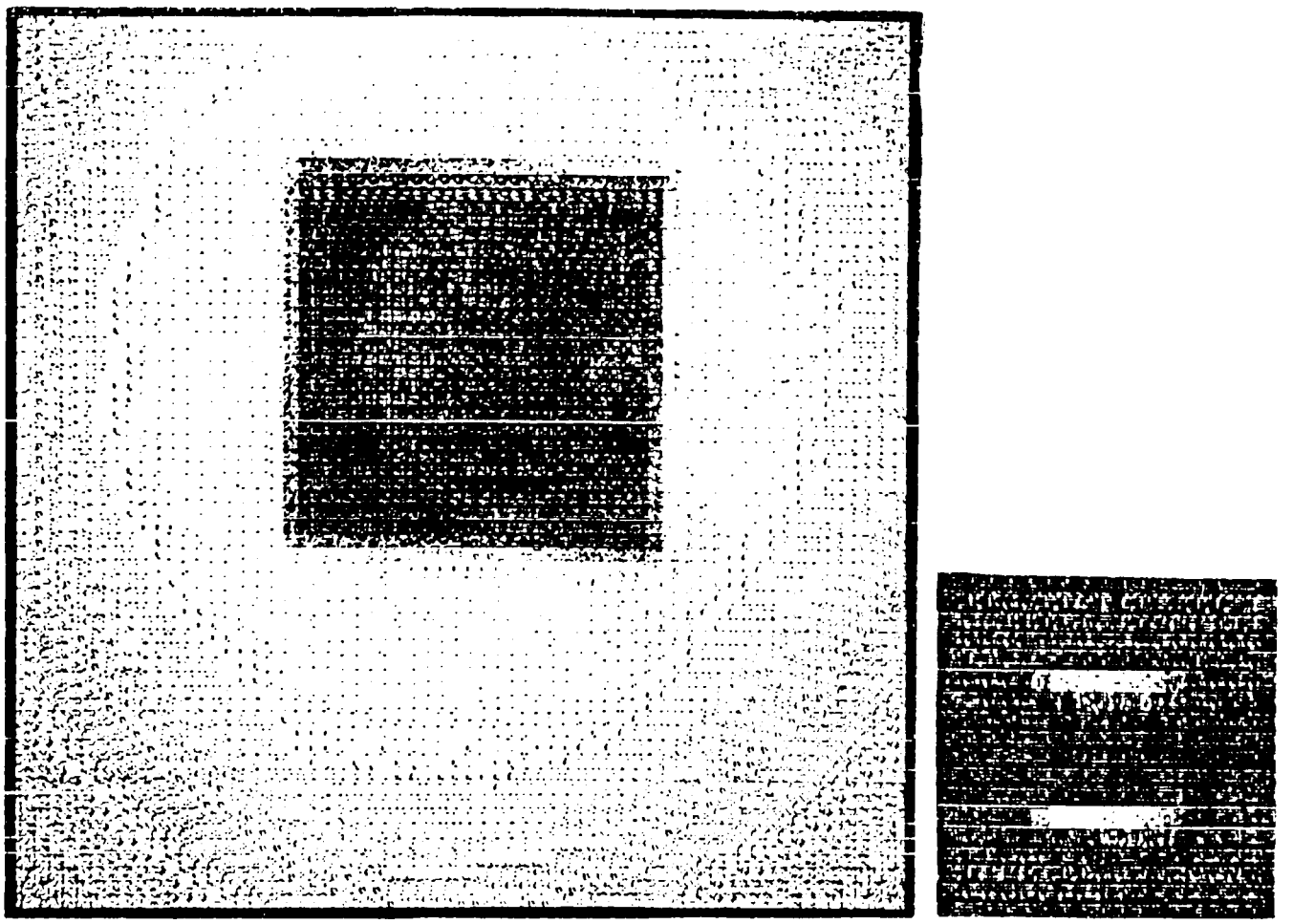
135°

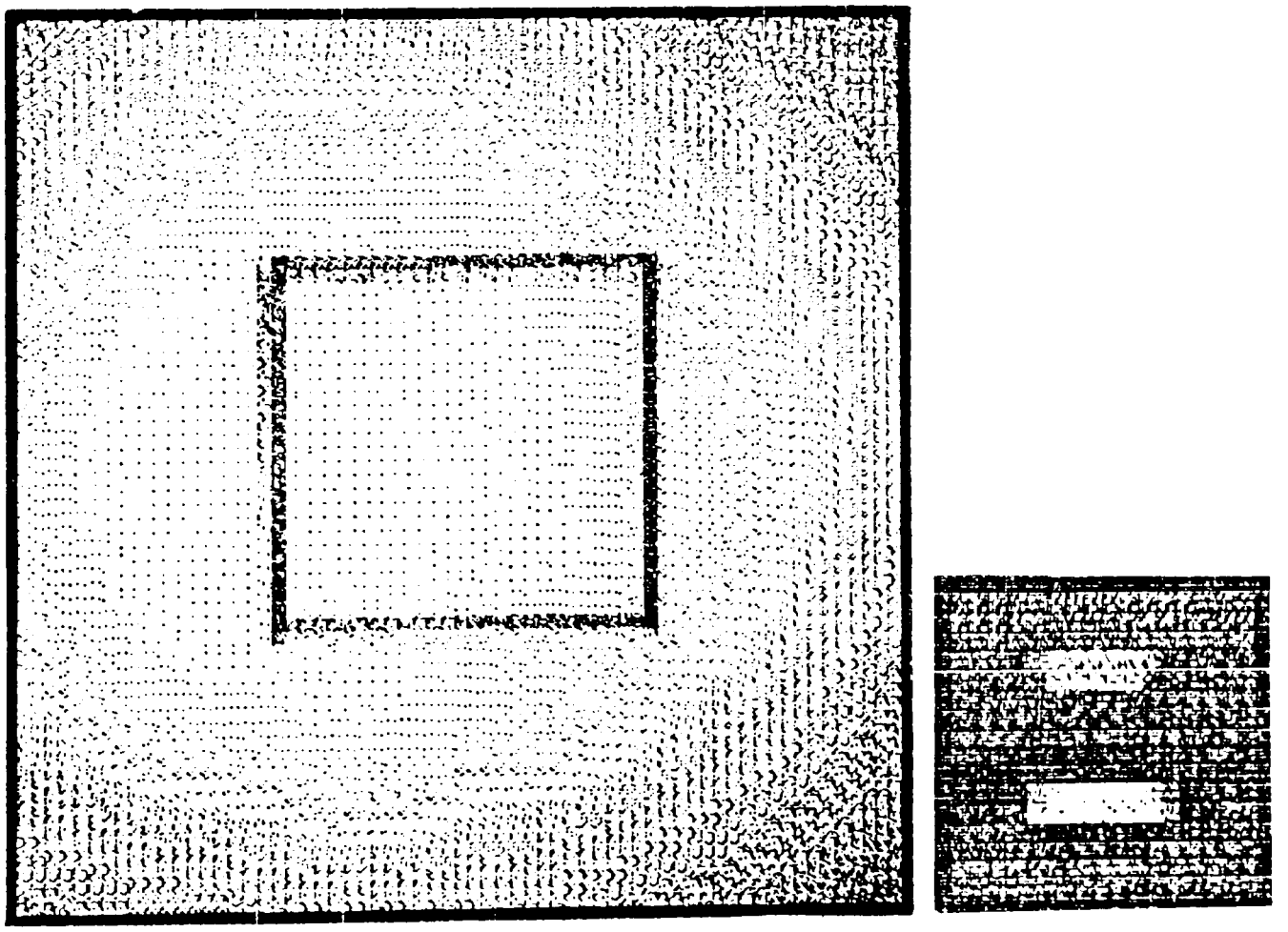


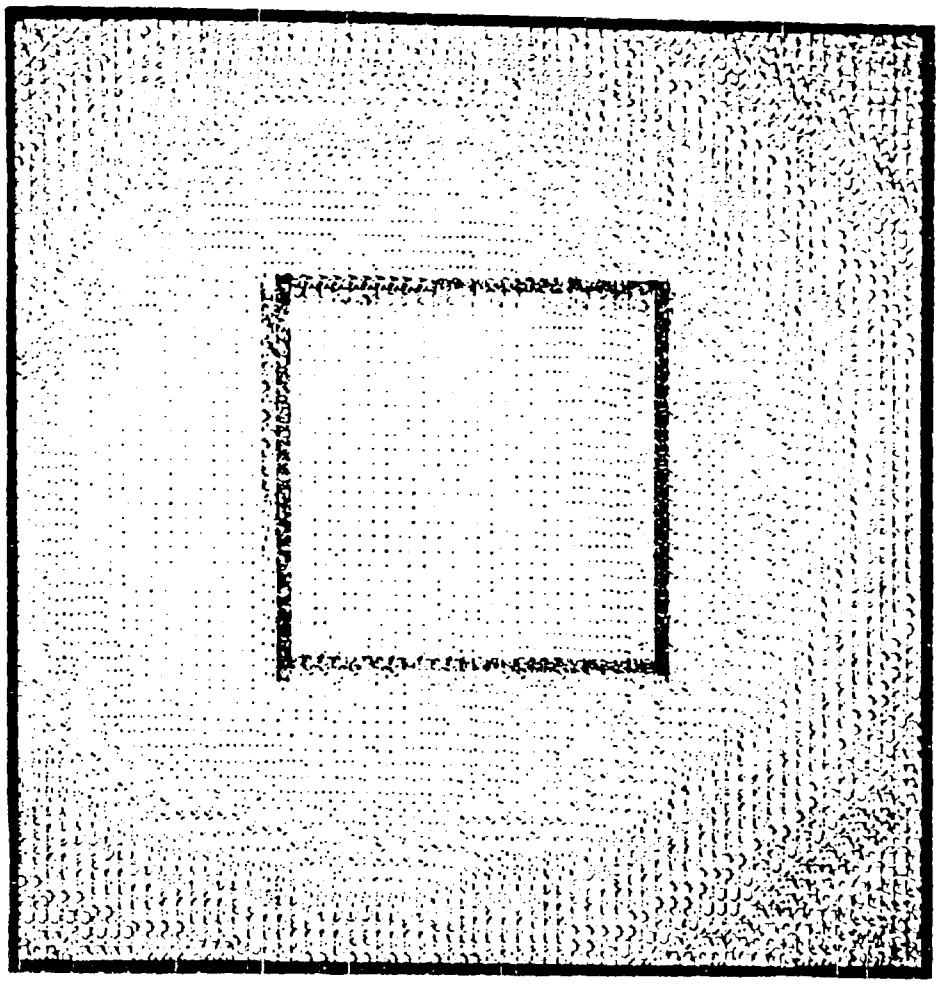
0°

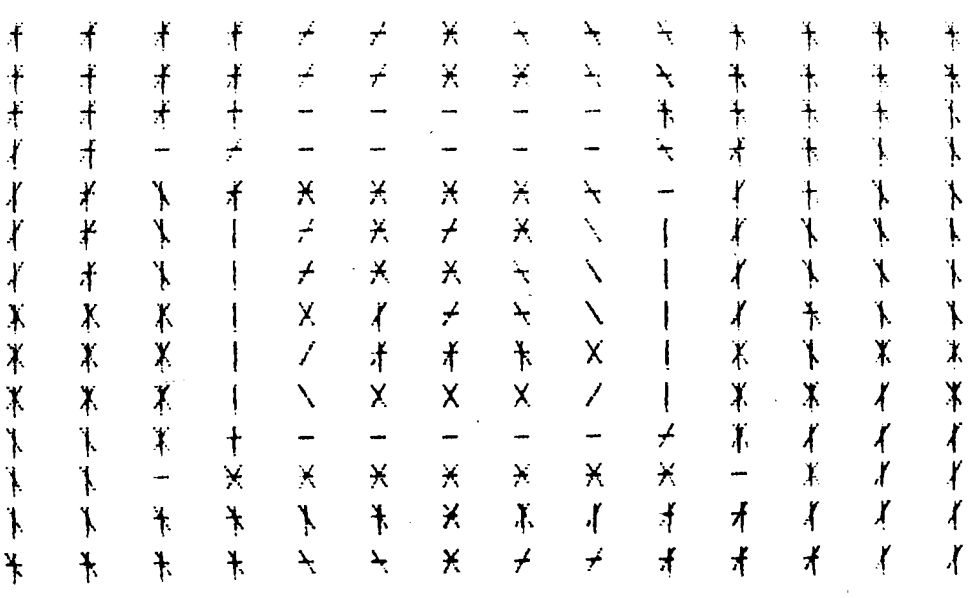
45°

90°

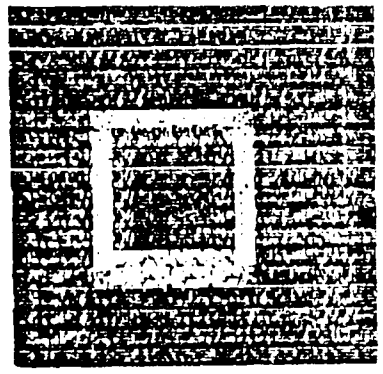








11	25	30	30	29	23	16	15	25	27	28	14	11	12	29	81
20	15	29	33	40	22	10	17	13	18	44	35	17	20	35	71
18	5	20	27	43	30	20	11	26	23	29	47	28	25	40	68
21	13	4	12	36	36	28	9	8	14	29	40	38	20	50	73
21	13	9	314	434	414	389	380	380	379	339	27	45	29	54	78
26	22	67	470	143	137	141	143	144	147	418	19	40	32	54	73
29	22	67	480	83	10	16	16	16	58	427	23	39	53	57	79
25	28	70	489	86	15	13	13	5	61	433	24	28	53	63	84
32	15	62	492	86	2	1	9	15	58	452	22	26	53	60	86
35	23	63	492	88	6	6	3	3	56	452	20	33	55	66	95
21	25	63	504	88	1	6	11	14	63	453	24	38	59	72	93
17	23	61	495	344	329	326	318	313	297	439	27	24	55	84	95
41	49	27	224	338	333	328	332	327	333	229	27	25	43	88	98
33	58	35	17	13	14	12	26	17	10	22	33	27	46	88	100
21	54	38	19	4	1	5	17	20	15	30	24	14	42	85	88
24	51	41	20	5	2	4	16	20	12	22	21	10	35	72	94

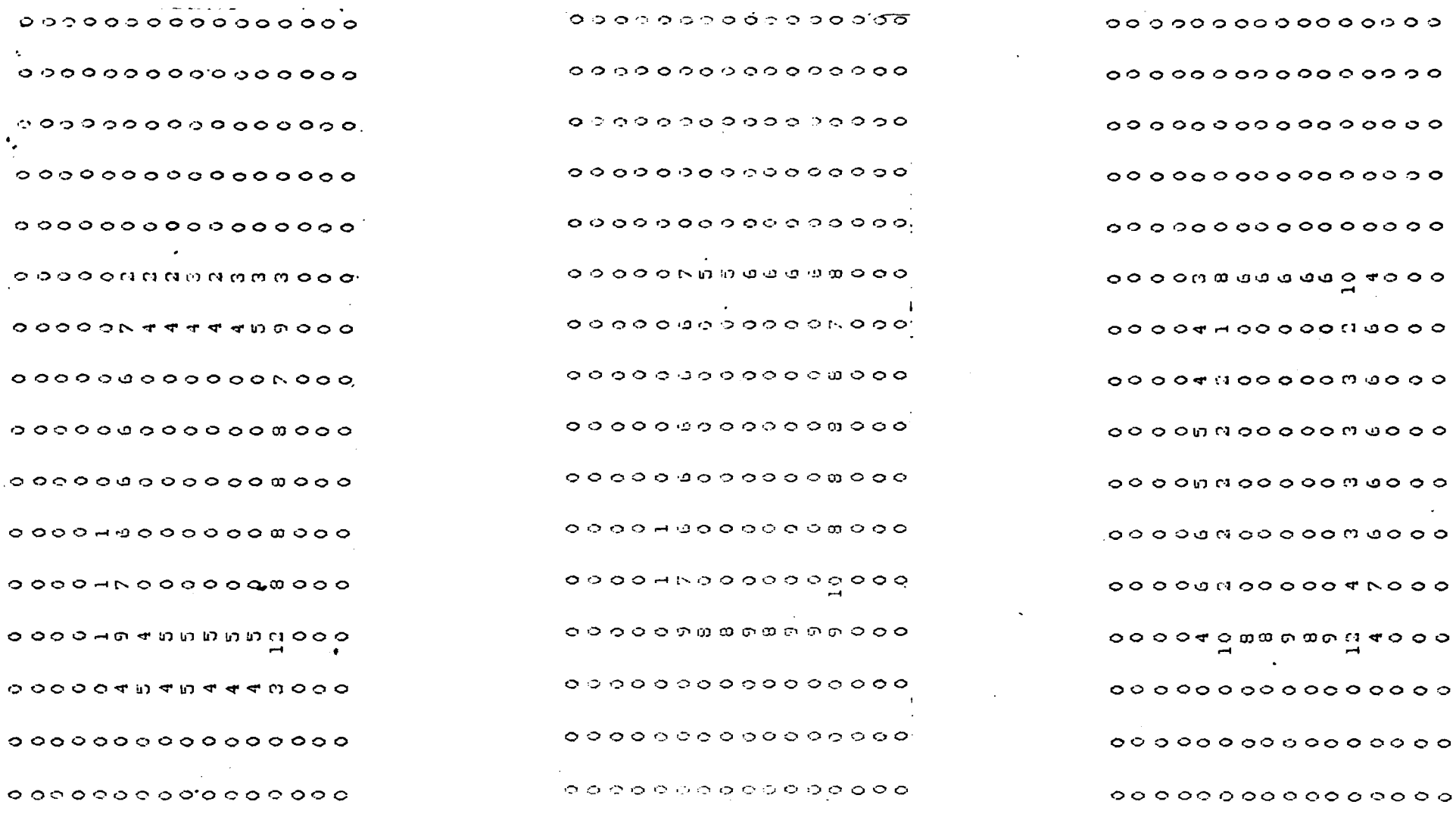


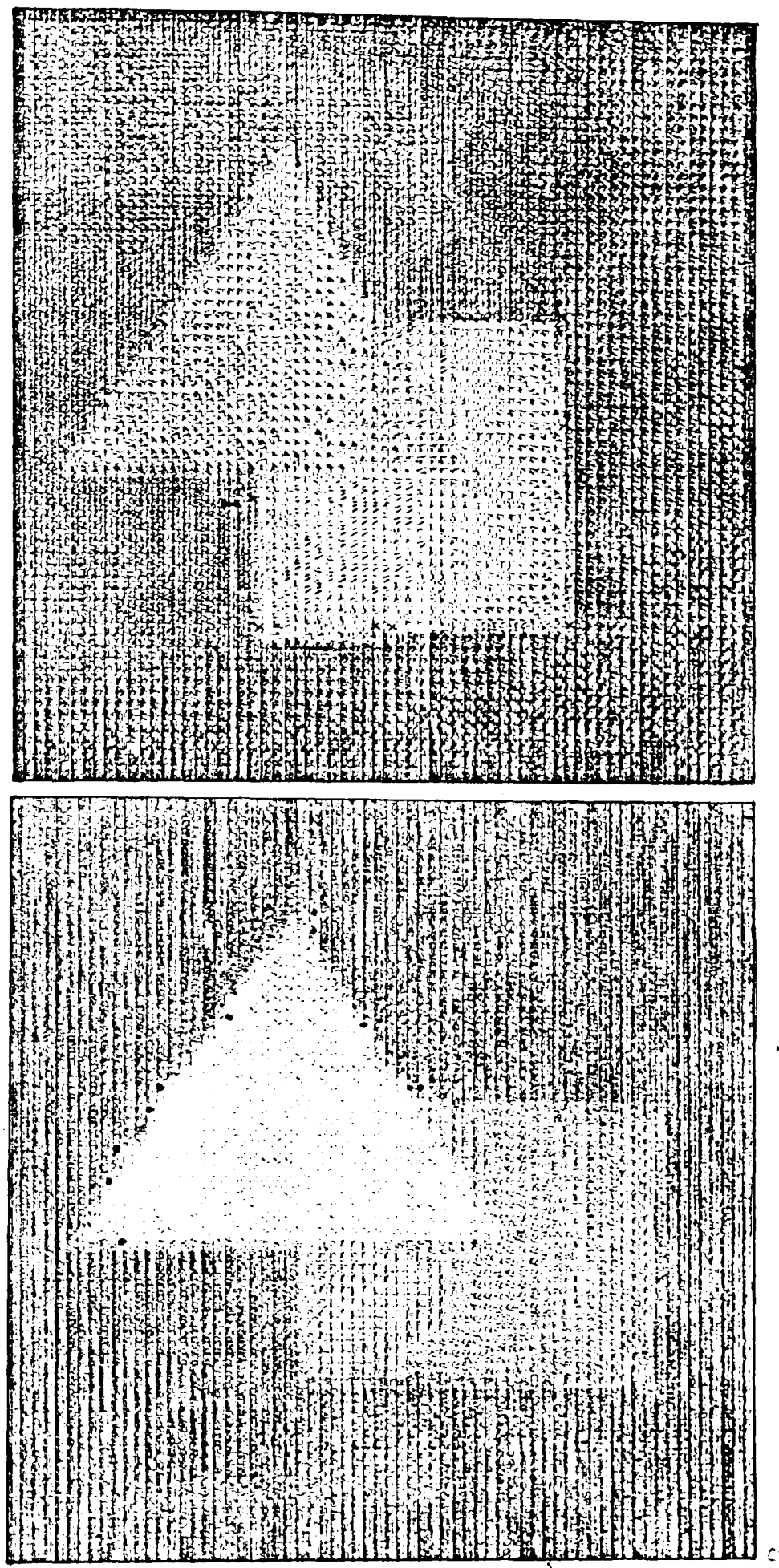
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	3	3	3	3	2	2	2	0	0	0	0	0	0
0	0	1	8	5	4	4	4	5	6	0	0	0	0	0	0
0	0	1	7	0	0	0	0	0	2	4	0	0	0	0	0
0	0	1	7	0	0	0	0	0	2	4	0	0	0	0	0
0	0	1	7	0	0	0	0	0	2	5	0	0	0	0	0
0	0	1	7	0	0	0	0	0	2	5	0	0	0	0	0
0	0	1	8	0	0	0	0	0	2	5	0	0	0	0	0
0	0	1	9	0	0	0	0	0	1	5	0	0	0	0	0
0	0	0	10	8	8	8	8	7	9	4	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

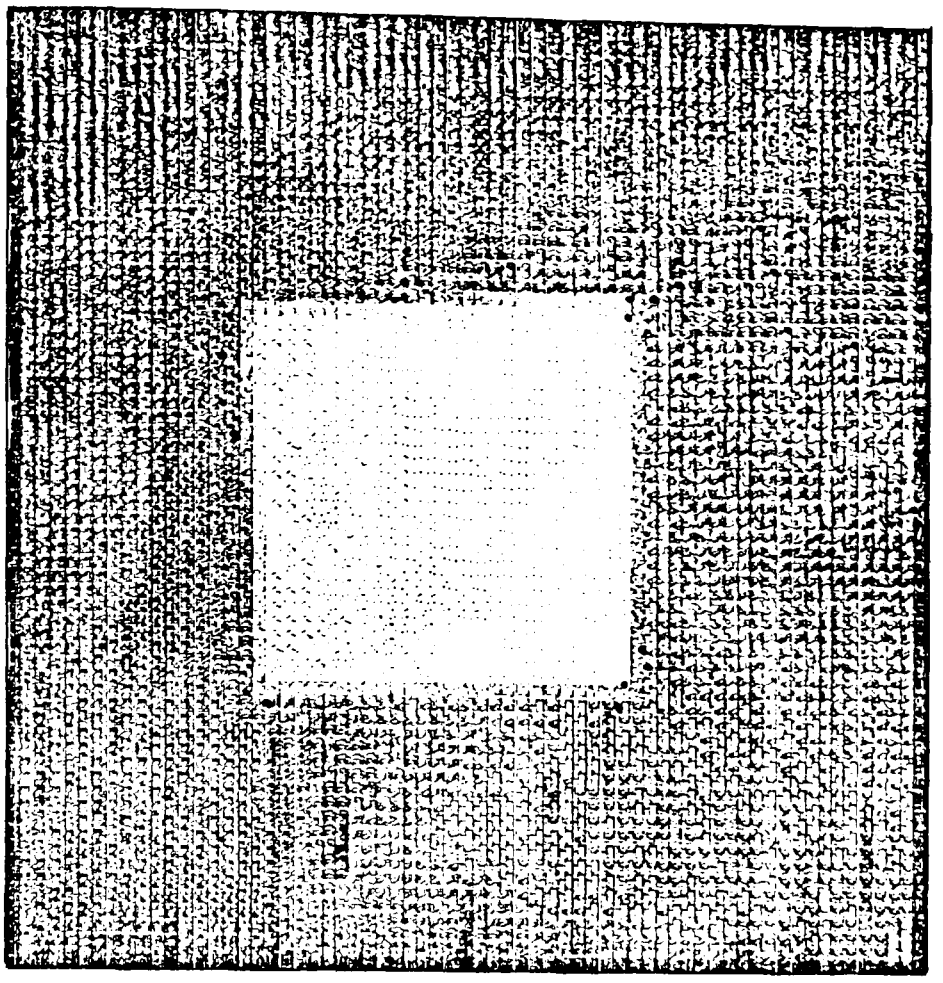
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	4	4	0	0	0	2	2	0	0	0	0	5	1	1	0
0	3	5	2	0	0	1	2	0	0	0	3	8	4	1	0
0	0	1	3	0	0	2	2	0	0	2	6	3	0	1	0
0	0	1	2	1	0	1	2	0	1	4	3	0	0	1	0
0	0	0	1	5	2	1	2	1	3	2	0	0	0	1	0
0	0	0	0	2	2	2	1	1	0	0	0	0	0	0	0
0	3	3	3	3	4	9	11	3	2	2	2	1	1	0	0
0	5	6	7	8	10	11	11	16	11	13	15	14	14	14	0
0	0	0	0	1	1	9	9	2	1	1	2	2	3	5	0
0	0	0	0	0	2	1	2	3	0	0	0	0	0	0	0
0	0	0	1	2	1	2	2	1	4	1	0	0	0	0	0
0	0	1	3	2	0	2	2	0	2	6	1	0	0	0	0
0	0	3	2	0	0	2	3	0	0	2	4	1	0	0	0
0	3	1	0	0	0	2	2	0	0	0	1	3	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

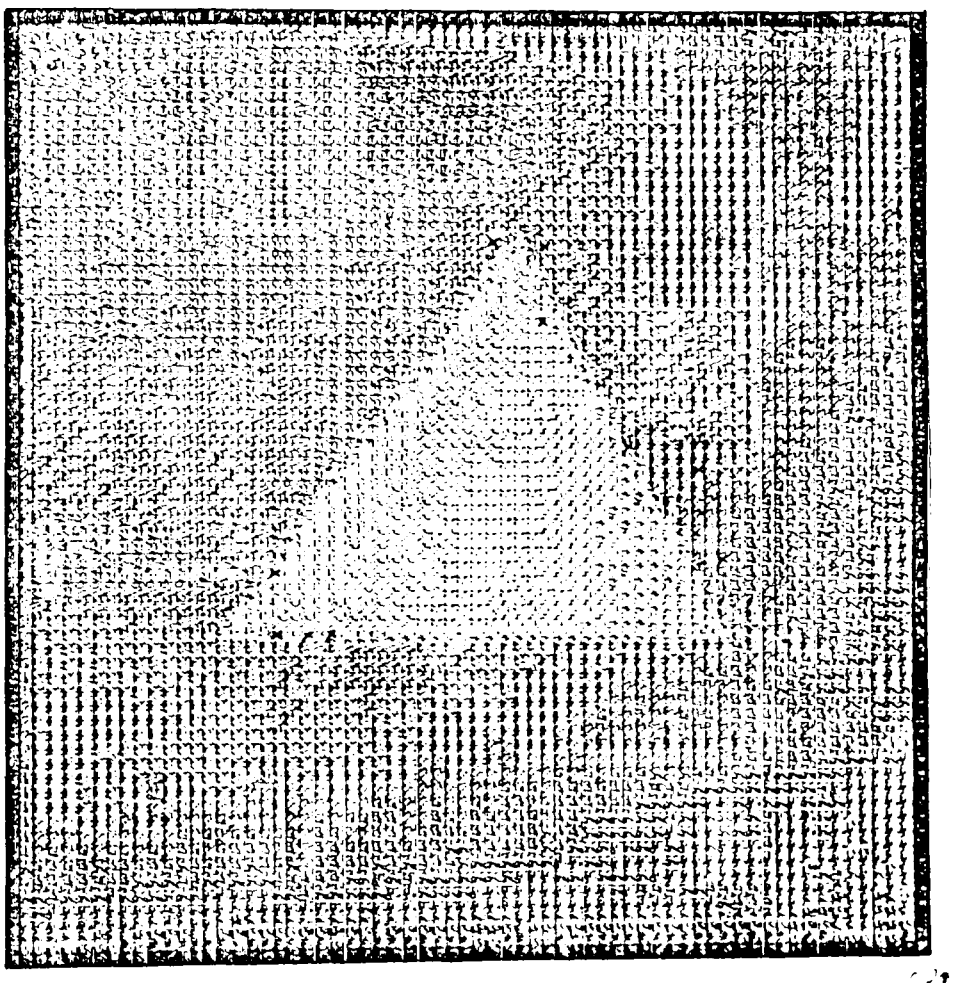
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	4	12	3	0	0
0	0	0	0	0	0	0	0	0	0	0	12	10	2	1	0
0	0	0	0	1	4	5	2	0	0	0	11	3	3	1	0
0	0	2	50	73	58	44	15	16	3	3	18	2	1	1	0
0	0	52	44	12	11	11	1	5	16	16	20	14	24	7	0
0	24	61	4	12	6	1	2	0	0	2	9	13	18	19	0
0	74	35	8	10	1	0	4	0	1	1	0	5	17	31	0
0	79	5	13	1	1	2	3	1	1	1	1	1	3	8	0
0	47	2	7	4	1	2	3	4	1	0	2	1	10	3	0
0	58	12	2	6	4	3	3	1	1	0	1	12	6	6	0
0	50	43	0	2	2	3	1	2	1	12	6	6	8	10	0
0	17	61	29	0	0	2	1	4	20	30	33	0	0	0	0
0	0	8	63	51	32	65	45	53	37	1	20	6	1	0	0
0	0	0	2	35	40	23	11	6	3	2	12	6	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

op11









: Teaching Procedure

START

Process the low resolution image for areas of 'interest'.
Rank areas of interest depending on detector response, max. responding area at the top of the list.
Set Q = 0

Move to area of interest Q in list where Q = 0 is the area of most interest.
Test 'picture to class' memory.
Save class recovered in class1.

Test 'class to picture' memory on class1.

Is the hamming distance between the recalled picture and the input picture greater than x ?

NO

YES

Create a new N point class pattern.
Teach 'picture to class' memory on the input pattern and the new class.
Teach the 'class to picture' memory on the input image and the new class.

Teach the 'class to picture' memory on class1 and the input picture.

Is Q set to the last interest area in the list ?

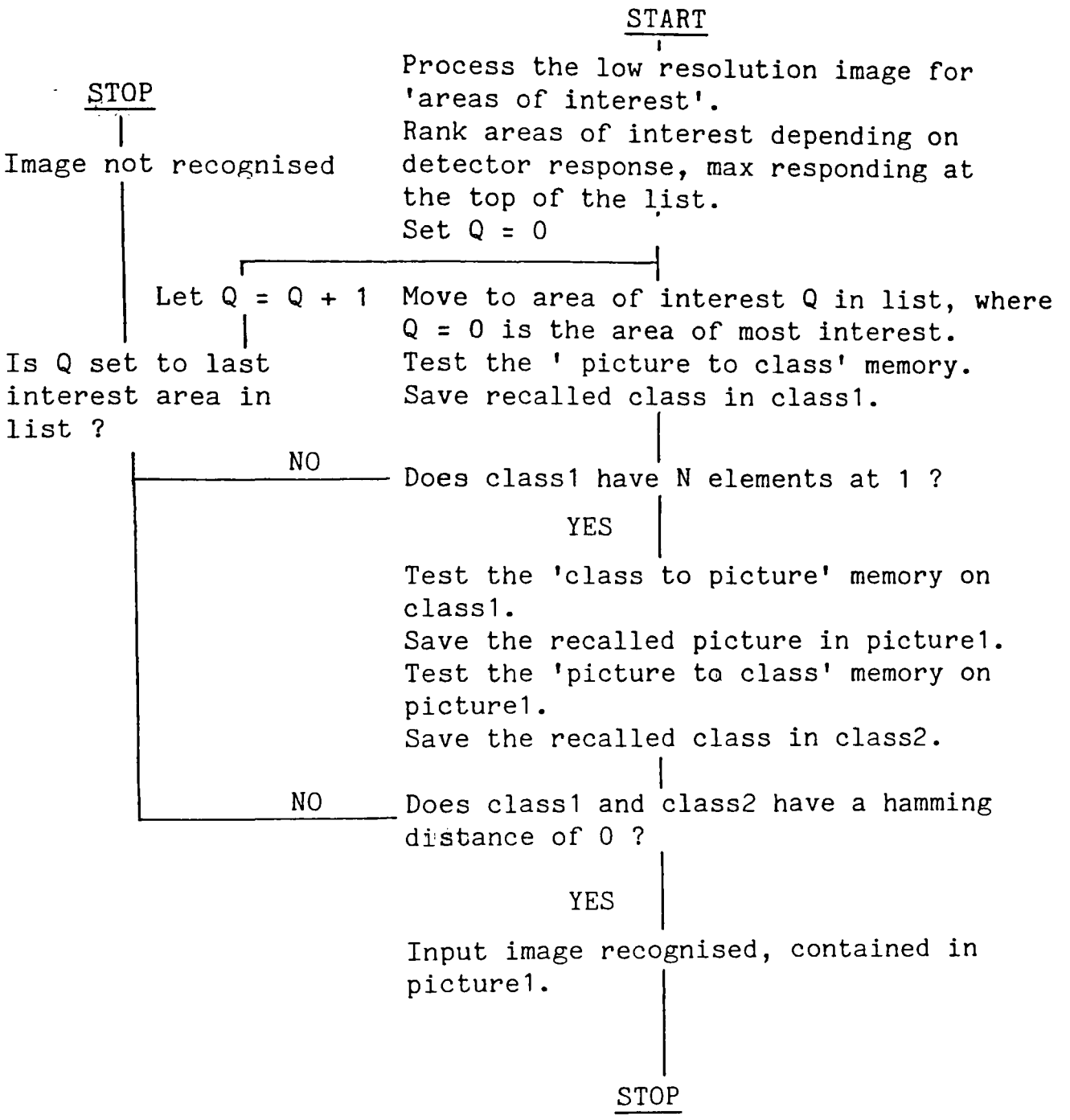
NO

YES

Let Q = Q + 1

STOP

Flow chart 2 : Reconition Procedure



NOTES

The input picture is N tuple processed in one of the ways described in chapter 9 before each teach or test.

References to 'picture' relate to the N tuple state data.

References to 'class' relate to the N point thresholded class.

Table A - Chapter 7

Efficiency figures for P -> C memory

Input type	Picture Size	Images Stored	Listing Mem	Associative Mem.	efficiency factor
Confidence	1024	5000	640000	125000	5.12
"	2048	7000	1792000	250000	7.17
"	4096	9000	4608000	500000	9.2
Random	16384	12000	23 x 10 ^ 6	2 x 10 ^ 6	11.5
"	65536	16500	1.3 10 ^ 8	9 x 10 ^ 6	14.4
"	262144	21500	7.0 10 ^ 8	32 x 10 ^ 6	21.8

NOTES

Input type : Type of input processing used on input image.

Picture size : Size of key vector to the memory in bits.

Images Stored : The Number of images stored.

Listing Mem : Memory needed to store the number of images in a listing memory, given in bytes = Picture size x images stored.

Associative Mem. : Memory needed to store the images in the associative memory. Figures are calculated from the equations given in chapter 7 with probability of error set to 0.001 and class points set to 5 for all calculations. Results in bytes of storage.

Efficiency factor : Storage needed to store patterns in a listing memory / storage needed in an associative memory.

Table B - Chapter 7

This compares the storage efficiencies of the two associative memories.

Class Size	Images Stored (P->C)	Images Stored (C->P)	Picture points	Efficiency (P->C)	Efficiency (C->P)
A					
32	40	20	64	1.2	0.6
64	82	41	64	1.2	0.6
128	164	82	64	1.2	0.6
B					
32	63	74	16	1.9	2.3
64	126	139	16	1.9	2.3
128	251	261	16	1.9	2.3

NOTES

Picture array size = 896

Number of picture points set to one = as in table

Class array size = as in table

Number of class points = 18

Probability of error = 0.001

Images stored is the number of images stored by each memory, this is derived from equation number 3. chapter 7.

The efficiency figure is given by

$$\frac{\text{Conventional file store memory use}}{\text{Associative memory storage}} = \frac{\text{Images stored x Size of image}}{\text{Class array size x picture array size}}$$

TABLE 1a

Test number	9	8	7	6	5	4	3	2	1	0
Identity	ânone	ânone	ânone	X	X	X	ânone	X	X	X
Image out	.	.	.	.	.	.	.	.	.	.
Response in	36/91	39/13	37/116	52/104	47/105	37/93	37/112	51/84	46/97	43/116
Class analysis	88/14	90/11	89/13	90/19	88/12	89/16	89/16	90/11	87/15	94/13
Worst case	50	43	48	41	52	46	36	58	33	43
cl. p. 1	4	4	4	4	4	4	4	4	4	4
cl. p. 2	4	4	4	4	4	4	4	4	4	4

TABLE 1b

Test number	9	8	7	6	5	4	3	2	1	0
Identity	X	X	X	X	X	X	X	X	X	X
Image out	B	B	B	B	B	B	B	B	B	B
response in	36/1	39/0	37/0	52/0	47/0	37/0	37/1	51/0	46/0	43/0
class analysis	88/6	90/7	89/6	90/14	88/8	89/9	89/13	90/4	89/11	94/8
worst case	66	44	59	44	57	63	37	69	38	51
cl. p. 1	4	4	4	4	4	4	4	4	4	4
cl. p. 2	4	4	4	4	4	4	4	4	4	4

TABLE 1a and 1b Results for investigation 1. Chapter 12

1a) Results for a system taught on non thresholded N tuple data, N tuple confidence data used during recognition. See text for a description of the results.

1b) Results for a system taught on thresholded N tuple data. N tuple confidences used during recognition. (* => case where the identity image was saturated).

TABLE 2

Results for investigation 2, chapter 12.

These results are for the tests listed in chapter 12 when applied to a system which was taught on thresholded N tuple data. See following pages for a description of variables.

Experiment	A	B	E	F	C	D	G	H
Test arrangement								
Input image thresholded	n	n	y	y	n	n	y	y
Confidance used	y	n	y	n	y	n	y	n
Image tested	tr	tr	tr	tr	sq	sq	sq	sq
No of identities with + recovered	9	7	9	8	2	1	2	2
No of identities with X recovered	0	0	0	0	5	3	5	5
Image recovered:triangle	6	2	7	5	1	1	1	1
Image recovered:square	0	0	0	0	4	3	4	3
Ave. of worst case when image OK	16.0	4.5	7.3	4	10.7	2.6	11	7
Ave of worst case when image wrong	3.7	1.8	3.3	1.1	2.5	1.7	1.6	1.5
Av. class < thresh, OK	53	30	53	13	3.5	25	34	19
Av. class > thresh, OK	11	1	11	2	6.5	6.3	5.6	4.6
Ave. of Im. and resp.OK	15	19	13	11.8	16	18	14	17
Ave. Resp not Im. OK	26	33	26	30	31	30	27	31
Ave. of Im. and resp. WRONG	5.8	2.5	1.3	3.2	4.6	4	4	5.3
Ave. Resp not Im. WRONG	7.2	9.6	8	6.6	14.0	8.7	15.0	12.0

TABLE 3

These results are for investigation 2 in chapter 12, when applied to a sytem which was taught on non thresholded N tuple data. Some results from table 2 are included for comparison. See the followin pages for a description of the results.

Experiment	A	B	E	F	C	D	G	H
Test arrangement								
Input image thresholded	n	n	y	y	n	n	y	y
Confidance used	y	n	y	n	y	n	y	n
Image tested	tr	tr	tr	tr	sq	sq	sq	sq
1) Non Thresholded teach data results								
No of identities with + recovered	2	2	3	5	0	.	0	0
No of identities with X recovered	1	0	0	0	1	.	0	0
Image recovered:triangle	.	.	.	.	.	.	.	.
Image recovered:square	.	.	.	.	.	.	.	.
Ave. of worst case when image OK	6.5	1	2.7	1.2	8.9	.	5.1	1.8
Ave of worst case when image wrong	1.9	3.5	1.0	2.5	2.4	.	1.7	1.8
Av. class < thresh, OK	71	35	68	15	55	.	42	20
Av. class > thresh, OK	26	12	22	6	17	.	13	6.4
2) Thresholded teach data results from table 2								
No of identities with + recovered	9	7	9	8	2	1	2	2
No of identities with X recovered	0	0	0	0	5	3	5	5
Image recovered:triangle	6	2	7	5	1	1	1	1
Image recovered:square	3	0	0	0	1	0	1	0
Ave. of worst case when image OK	18.0	4.0	7.0	1	10.7	2.6	11	7
Ave of worst case when image wrong	2.7	1.3	3.3	1.1	2.5	1.7	1.6	1.3

Teach cycle	A			B			C			D			E		
	i	ii	iii	i	ii	iii	i	ii	iii	i	ii	iii	i	ii	iii
0	9468	852	5123	8731	1319	3872	9020	651	6973	8949	1365	3765	8949	987	6334
1	9468	852	5123	8731	1262	3872	9020	766	6913	8949	1365	3765	8949	1014	6334
2	9468	852	5123	8731	1327	3872	9020	867	6913	8949	1365	3765	8949	1113	6334
3	9468	879	5123	8731	1338	3872	9020	984	6803	8949	1365	3765	8949	1203	6334
4	9468	1043	4535	8731	1338	3872	9020	952	6803	8949	1394	3765	8949	1294	6334
5	9468	1132	4535	8731	1350	3872	9020	991	6803	8949	1435	3765	8949	1348	6129
6	9468	1379	4535	8731	1398	3872	9020	991	6803	8949	1486	3765	8949	1362	6129

Picture to class test results.

Teach cycle	A			B			C			D			E		
	i	ii	iii	i	ii	iii	i	ii	iii	i	ii	iii	i	ii	iii
0	0	43	4	0	46	4	0	51	4	1	37	4	0	37	4
1	0	43	4	0	46	4	0	51	4	2	37	4	0	37	4
2	0	43	4	0	46	4	0	51	4	2	37	4	0	37	4
3	1	43	4	0	46	4	0	51	4	2	37	4	0	37	4
4	1	43	4	0	46	4	0	51	4	2	37	4	0	37	4
5	1	43	4	0	46	4	0	51	4	3	37	4	0	37	4
6	1	43	4	0	46	4	0	51	4	3	37	4	0	37	4

Class to picture test results

TABLE 4a : Test on the square.

Teach cycle	A			B			C			D			E		
	i	ii	iii	i	ii	iii	i	ii	iii	i	ii	iii	i	ii	iii
0	6461	1196	2455	6570	916	4249	6517	691	4380	5865	584	4092	6618	1321	2557
1	6461	1230	2455	6570	976	4177	6517	653	4380	5865	634	4092	6618	1336	2557
2	6461	1263	2455	6570	1103	3979	6517	670	4380	5865	654	4092	6618	1354	2557
3	6461	1283	2324	6570	1138	3979	6517	670	4164	5865	677	4092	6618	1254	2557
4	6461	1298	2324	6570	1264	3504	6517	505	4164	5865	736	4092	6618	1254	2557
5	6461	1372	2324	6570	1291	3504	6517	754	4164	5865	763	4092	6618	1369	2557
6	6461	1437	2324	6570	1327	3504	6517	823	4164	5865	775	4092	6618	1401	2557

Picture to class test results.

Teach cycle	A			B			C			D			E		
	i	ii	iii	i	ii	iii	i	ii	iii	i	ii	iii	i	ii	iii
0	6	39	4	3	41	4	1	40	4	1	35	4	0	40	4
1	6	39	4	3	41	4	1	40	4	1	35	4	0	40	4
2	8	39	4	3	41	4	1	40	4	1	35	4	0	40	4
3	8	39	4	3	41	4	1	40	4	1	35	4	2	40	4
4	8	39	4	3	41	4	2	40	4	1	35	4	3	40	4
5	8	39	4	3	41	4	3	40	4	1	35	4	3	40	4
6	8	39	4	3	41	4	3	40	4	1	35	4	3	40	4

Class to picture test results.

TABLE 4b : Test on the triangle.

Description of variables from tables 2 and 3.

Experiment :

Individual runs on the taught system, each with a different set up as given by the 'test arrangement'.

Input image thresholded :

Whether or not (y or n) the input N tuple data was thresholded prior to testing.

Confidence used :

Whether or not (y or n) confidences were used during memory recall.

Image tested :

Which shape the system confined its window positions to - tr = triangle - sq = square.

No of identities with + recovered : No of identities with x recovered :

How many tests resulted in the recovery of a '+' or a '-' identity pattern.

Image recovered:triangle : Image recovered:square :

The number of tests that resulted in the recovery of a square or a triangle.

Ave. of worst case when image OK :

The average worst case class distance (class confidence) when the image recovered was that expected by predictions (see text).

Ave of worst case when image wrong :

The average worst case class distance (class confidence)

when the image recovered was that not expected by predictions (see text).

Ave. class < thresh OK :

When ever a test resulted in the expected recall pattern the average response of the class points which were set to 0 was recorded. These values were averaged over all tests for each run and given in the table.

Ave. class > thresh OK :

As above but relating to the average response of class points which were set to 1 after N point thresholding.