INVESTIGATIONS INTO TRAINABLE PICTURE PROCESSING SYSTEMS

A thesis submitted for the degree of

Doctor of Philosophy

of the University of London

by

Martin Mayer

Department of Physics

Royal Holloway College

University of London

August 1982

M. MAYER

INVESTIGATIONS INTO TRAINABLE PICTURE PROCESSING SYSTEMS

ABSTRACT

This work concerns the development of a new type of picture processing system for images represented as digital arrays of pixels. This is a synthesis of two established ideas, already under independent investigation. The first of these is picture processing by look-up tables. This is a fast method of generating pixel outputs as a result of input pixels accessing a particular region of a look-up table, pre-loaded with the required data. The second idea is the use of RAMs as learning machines. Here, RAM elements are connected together so as to be alterable in data content by training stimuli in a coherent manner. This results in a system able to exhibit definite responses to later test stimuli, and thus identify these stimuli unambiguously.

The methods used for bringing these two concepts together are described here. A practicable picture processor results, which can be trained by examples. That is, it can perform a picture transformation simply by presenting to the machine (in a prior training phase) examples of the process. From this, the machine deduces the information necessary to be able to perform the same transformation on unseen patterns.

Experiments have been performed on a wide range of variations on this theme. Different types of machines acting on different data and tasks have been tried, under various conditions. A description is given of these machine variations, together with a generalized system for describing such variations more formally. The machines were simulated in practice on a microcomputer system. The simulation software used in these investigations is also described.

Finally, the implications and limitations of such machines are discussed with reference to their ultimate performance and possible applications in fields other than picture processing.

"Contrariwise," continued Tweedledee, "if it was so, it might be; and if it were so, it would be; but as it isn't, it ain't. That's logic."

Lewis Carroll

CONTENTS

CHAPTER 3

EXPERIMENTS WITH BASIC LPP MACHINES

CHAPTER 4

THE GENERAL LEARNING PICTURE PROCESSOR

CHAPTER 5

EXPERIMENTS WITH MORE ADVANCED LPP MACHINES

CHAPTER 1

LEARNING PICTURE PROCESSORS

1.1 Introduction

The advent of data processing that followed the development of the digital computer brought with it many entirely new disciplines. Amongst these is a set of applications that requires deductive and intuitive processing, related to the processes of reasoning and perception in the brain. One of the more important examples of this is concerned with visual image processing and classification, an area that has been ever more closely studied over the last two decades.

1.2 Applications

There are many applications of machines able to process and classify pictures, and several examples of these will be given here.

Optical character recognition is of great use in the world of business - many types of document formerly read by human operators can be read and subsequently processed entirely by machine (69,10). In medical applications, image analysis may take many different forms (75), including X-ray screening (34), cancer cell scanning (18) and radiograph analysis (36). These all exhibit characteristic patterns, enabling subsequent diagnosis of symptoms - also by machine (56). A combination of OCR and the medical field results in

practical blind reading aids (54).

Military and space applications are also under investigation (29), although such applications may be classified and consequently difficult to research. Automatic identification and tracking of objects is often the goal here, for the purpose of targetting weapons automatically (39). Space satellites send picture data in need of considerable processing in the form of filtering (eg. for noise removal) or other tranformations (eg. to remove distortions, blurring or compression and expansion for the extraction of the maximum information (44)). Video images (of any subject) can make use of filtering operations or even modifications for special effects. The present generation of video effects mixers used on broadcast television are examples of this. Security systems make use of automatic monitoring of CCTV systems for alarm situations, and subsequent matching and identification of faces or fingerprints that may be recorded (17,62).

Scientific laboratories can make use of machine scanning of signals (often comprising vast quantities of data) in a search for a particular pattern or feature (68,67). The type of features sought are often abstract, to the extent that even human observers can sometimes be unsure of identification.

Many industrial applications also exist - a robot assembly device can benefit from visual input (3,8,46). This must be suitably processed to give the relevant information to help it as a capable and, more important, general-purpose tool. This can be extended to transportation, where visual inputs to an autonomous vehicle can greatly aid its

movements in unknown territory (33). This can be applied across a wide range of complexities - from manoeuvering fork lift trucks to piloting aircraft.

There is obviously considerable scope in the choice of a problem to be tackled. Unfortunately this may give rise to a multitude of attempts at solutions - often 'tuned' to a particular application. While this undeniably extracts the maximum potential from a solution, many are left very specific and over-adapted to a particular application. They become no longer suited to a range of users. There are vast numbers of alternatives to be explored even within each attempt at a solution.

1.3 Alternatives in Picture Processing

At every level of the search for a picture processing solution, there are alternatives to be considered. This may slow the progress of research by directing it towards a 'tuned' solution, specific to a problem. It can also be cited as a reason for assuming that no general solution exists. It is in an attempt to generate at least a partially general picture processing solution that the work described in this thesis has been carried out. While it is necessary to make several specific choices before any form of picture processing can commence, it will be shown here that a single, simple machine can readily perform a relatively large range of tasks without further specialisation.

1.3.1 Alternatives in the Representation of Picture Data

The alternatives that exist in picture processing begin with the data. The data are in some form of organised, reproducible representation of visual images, with a large number of alternative storage methods available. Analogue storage has been used (37) in the days before the advent of cheap digital devices, yet today the latter method is used almost exclusively.

Digital pictures are most often represented as two-dimensional arrays of picture elements (pixels). The number of elements and their spatial relationship are some of the first alternatives to be specified in such picture processing machines. Once the number of pixels per picture and the tessellation have been defined, the quantity of information per pixel is be decided. The representation can vary from simple binary (black and white) to a full range of colours at each point. Subsequently, if the picture is to represent a moving image, a frame repetition rate must be specified. Frequently, a single static picture is used in experimentation, but ultimately many of the applications mentioned above would require the use of moving pictures. This is for two reasons:

1 many pictures change in time (which must be recorded),
2 much potentially useful information can be gathered from the manner of these changes.

Normal TV video rates are often used as a standard, to suggest some values for these variables. This rate is equivalent to approximately 12 bits per pixel, 625 by 833 pixels per frame and 25 frames per second for a full colour,

moving television quality image. This is equivalent to a gross data rate of $\sim 10^8$ bits per second, and is cited as a goal for many current devices (78). Alternative data rates are well-known and documented in many books on the subject (7,15).

1.3.2 Global or Local Processing of Data

Broad alternatives also exist in the processing carried out on the data. The choice to be made is between local and global approaches. In the former case, processed areas of the picture are solely a function of the local area in the picture before processing. Global processing allows any processed picture area to be a function of any or all of the regions in the unprocessed picture. Global operations lend themselves well to transform processing - possibly by optical means (47). At present this method has enormous advantages in speed over discrete processing, being effectively instantaneous. However, when implemented on serial processors, global processing is necessarily much slower than local processing. This is the predominant factor accounting for the emphasis on the latter in much of the image processing literature.

1.3.3 Sequential or Parallel Processing

In the case of local processing of discrete pixel arrays, there exists the alternative of parallel or sequential processing (63). In the former, all localities are processed simultaneously (either actually or effectively). Each resultant new local area is created in a

different picture space from the original picture. Several hardware devices have been created with architecture to implement this type of processing (16,26,28).

In sequential processing, each locality is processed to finality before the process is applied to the next area in an ordered scan. The processed results are inserted back into the original picture space. This results in the input to the processor being part processed, part unprocessed pixels. This has important implications in many types of picture processing where topological variables such as connectedness are involved (70).

1.3.4 The Choice of the Size of the Locality

The size of the locality or neighbourhood mentioned above is related to the overall picture size and the number of pixels within it. This 'window' size is a major factor in such processing and is chosen subject to two constraints: with a large window there is the potential for more powerful processing, yet the smaller the window the faster and simpler the process.

1.3.5 The Choice of Algorithms

The final choice, and possibly the most important of all in picture processing is that of the actual algorithm employed. At this point it is interesting that a distinction can be drawn between an underlying strategy and the device-dependent algorithm, giving rise to yet further alternatives in approach.

Here is considerable scope for what is often a rather arbitrary variation. Algorithms are generated by introspective analyses of how the process should be done, and thereafter evaluated on the results of their application. Comparisons between man and machine have been made (57), which are inclined to suggest man's methods as a suitable, if not the only source of solutions. Consequently, this is the area of picture processing most subject to 'ad hoc' solutions, and is thus where the main thrust of this thesis is aimed. A successful search will be made for a method of generating algorithms not based on such 'ad hoc' solutions. This will rely on examples of processed pictures alone, rather than premeditated methods of achieving these processed pictures.

1.4 Basic Problems in Picture Processing

The alternatives above lead to considerable problems in finding picture processing solutions. These may be summarized generally into three aspects:

1 there are vast quantities of information to be stored and processed,
2 the time taken to process pictures in real time must be controlled,
3 the processes to be executed are not known.

The ideal solution sought here will have to be fast, efficient and capable of generating its own algorithms.

The problem of speed cannot be overlooked in many applications which, once divorced from the laboratory, must

ultimately work in real time. This requires efficient algorithms capable of working quickly to generate outputs from inputs: yet even this is usually insufficient. Special purpose hardware is currently necessary to approach the data processing rates required. Early attempts at reducing the execution times of processors resulted in parallel machines (79,80) as an approach to speeding up the hardware. The current generation of programmable general purpose computers is still far too slow to cope with the video data rates cited above as a standard. The hardware that must be produced is often grossly restricted in its adaptability to even closely related problems. This may result in forced simplifications of the algorithms that can be implemented in the chosen application area.

Look-up tables are examples of such systems used for pixel value transformations (1). Shift register delay lines are examples that allow the processing of windows of pixels that are spatially adjacent, but temporally distant in serial data streams (42,74).

The solution proposed here will be based on a novel use of look-up tables in a writing mode, as well as a reading mode.

Apart from speed, the other major problem is that the algorithms required are unknown. That is, while the results required of the picture processor can be defined (perhaps by examples) there are no clues to the methods of solution. Attempts at finding the human methods employed here (31,71) have not been easily translatable into machine form. The usual approach is to break down the process into a set of 'primitives' - that is, a set of relatively well-defined

tasks capable of at least some successful intuitive solutions. These range from the trivial (such as inversion, or the removal of 'salt and pepper' noise) to the complex (such as thinning or object identification). These are often combined in a particular sequence to produce the overall image transformation required for a particular application.

Laudable attempts have been made at organising the tools to search for these algorithms. Examples of these are the VICAR software package developed at JPL, California (15); and the SUSIE package at Southampton (9). These command type languages relieve the researcher from the mundane aspects of picture processing, allowing concentration on the actual image processing techniques under investigation. A similar type of language has also been developed for pattern recognition: JANSYS at Brunel University (55).

However, solutions proposed as a result of these and other methods are further complicated by the choices highlighted in Section 1.3 above. This often confuses the pursuit of a good solution. Good proposals can be lost in an unfortunate combination of parameters (such as size, type of picture, etc.) thus obscuring the way ahead. As Groh stated in 1978 (32), it is for these reasons that "technical picture processing is still in its infancy".

In an attempt to avoid this arbitrariness in the cycle from problem definition to problem solution, the work in this thesis is proposed. Indeed, the broad range of solutions covered in several recent review papers (64,81) illustrates the fact that no genuinely 'best' strategies have crystallized.

1.5 The Need to Avoid 'ad hoc' Development

An example of where this 'ad hoc' nature of the art has resulted in an inefficient collection of algorithms is in the task of thinning. Thinning here is defined as the successive erosion of the edges of a figure until a unit width, connected skeleton remains along the figure's limbs (22). The algorithms that have been generated in the past (70,10) are not generally guaranteed to work. They may break the figure, or not discriminate between noise and limbs, or simply produce inaccurate skeletons. This record of bad performance has stimulated the generation of new, complex or arbitrary algorithms; but unfortunately these are not derived evolutionarily from earlier solutions (43). There is a need for algorithms to be 'guaranteed' to do the job, and one aid to achieve this is to define a 'requirements specification' before attempting to find a solution to meet it. Thereafter any proposed solutions can be tested to ensure they meet these specifications.

Thinning has arisen as a non-trivial problem that has been continually attempted because of its potential use, and actual implementation, in many practical applications. It has been used in printed circuit board manufacture (59), inspection of fibres on air filters (24), fingerprint classification (60) and chromosome analysis (38). The requirements can be specified as a capability to simplify the image by a reduction in limb width for subsequent analysis or data compression purposes. The difficulty is that this is a global problem where long-range features have to be taken into account, and is thus not strictly solvable with the often proposed local solutions.

To define the problem explicitly, a set of requirements can be expressed about the required thinning algorithm.

1 It must transform the image to a unit width skeleton.
2 It must maintain connectedness.
3 It must retain any line ends that it locates.
4 It must modify the image isotropically.
5 It must complete its task in a reasonable length of time.

Algorithms have been developed under these conditions, yet still found lacking, by breaking limbs (23), being over-complex and hence slow, or having other shortcomings. Although it is somewhat artificial to separate these properties in this manner, this can focus attention on the minimum requirements. Algorithms are usually proposed as complete solutions - and failure to identify sub-tasks or constraints properly is often the fundamental problem. Indeed, an attempt to meet all the requirements simultaneously usually results in not meeting some at all. However, acceptance criteria are highly desirable for quantifying the usefulness of a given algorithm. This may also tend to generate algorithms designed from the 'top down', with the advantage that they can be tailored to applications by re-adjustment of requirements at the design stage. Different tesselations, noise characteristics and noise levels are but three examples of variations that might invalidate a given algorithm. Ideally an algorithm should be readily modifiable to meet the new conditions. The only requirement is to produce a skeleton of predictable accuracy under given conditions. Until recently (22), there has been

no mention of such skeleton accuracy standards in the literature.

Davies and Plummer (22) have approached the problem with this methodology. A family of algorithms has been proposed that can be guaranteed to generate a skeleton rigorously defined as 'adequate'. These algorithms can be adapted to a range of conditions and thus are no longer 'ad hoc' solutions, but transportable between applications.

This structured approach to image processing is preferable - execution of a cycle of requirement specification, solution development, and validation. However, an alternative is proposed in this thesis, which even avoids the need to generate a solution by intelligent considerations. If the requirements can be specified formally, a device can in principle be created capable of generating its own algorithms directly from these specifications. The method used here for specifying the problem is the provision of examples of processed data. Additionally, this no longer restricts the task to thinning, or any picture processing task. The device is capable of performing any task that can be specified by the examples, and is not subject to any 'ad hoc' proposals of solutions. A device capable of such action will be described in the following sections.

1.6 RAMs as Processors in Learning Machines

It has been suggested that the present work will try to avoid any arbitrariness in finding solutions. Thus one of the most suitable sources of information available to define

the requirements is a set of examples. A picture set exists (or can be created) to illustrate the processes to be performed. The problem has consequently been modified to the creation of a machine capable of responding coherently to these examples, such that it can later copy the process onto new data. The example data themselves are used to drive the machine, to re-order it internally in such a way that it can reproduce the picture processing task on which it was trained.

RAMs have been identified as universal devices capable of emulating any logic circuit (2). In particular, RAMs have been examined in considerable depth by Aleksander et al. : as 'n-tuple' learning machines for pattern recognition (4,82,83). Such machines can be trained to respond to a pattern/class pair, such that they can later generate correct classes from unseen patterns (11,72,77). It will be shown here that a similar type of machine can be created that will react to a pair of examples, consisting of a raw picture input, together with a processed picture input. Such a machine will generate 'correct' output pictures from unseen inputs, and should fulfil the requirements above for avoiding arbitrary designs of algorithms.

The machine should be general in the sense that it will be:

1 task-independent,
2 capable of handling different tasks with a change of examples rather than re-programming,
3 able to handle different task complexities,
4 able to process different picture formats and
5 capable of generalisation.

This last point is of fundamental importance to any intelligent machine. In this context, the property of generalisation can be defined as the ability to process, correctly, pictures not previously seen as examples. For a learning picture processor to be of any use, this is vitally important. The question of whether the performance is 'sufficient' to be useful is to be investigated here.

1.7 The Proposed Learning Picture Processing Machine

The characteristics of the machine proposed herein are summarized below. This is an attempt at finding an optimum solution to producing an output picture data set, correctly transformed from an input picture data set. Assuming that examples of these data sets are available for training, a machine is proposed that can generate the required algorithm autonomously. This machine will be implemented in RAM devices used as modifiable look-up tables. This will be shown to result in fast operation - as seen elsewhere with this type of device (1,61,62). A practical version of the machine will be described, which makes use of currently available hardware technology.

1.8 A Summary of the Following Chapters

Chapter 2 will introduce the methods by which practical learning machines may be organized to process digital pictures, having learnt to do so only by example.

Chapter 3 will document preliminary practical experiments on such systems, and illustrate the results with

some proposals for modifications and improvements.

Chapter 4 will show a structured approach to the generation and categorisation of variations on the basic theme, and will propose a 'general machine'.

Chapters 5 and 6 represent the bulk of the experimental work which investigates a number of variations by experiment. The evaluation of the results leads to a convergence onto some general rules regarding such systems' behaviour.

Chapter 7 documents the simulation process used throughout these experiments, for creating and operating this range of learning picture processors on a conventional digital computer.

Chapter 8 will conclude by summarizing the implications of the systems proposed here, in the light of the current position of image processing; and also make some specific suggestions for further work. Some rather more general conclusions will also be drawn regarding the applications of learning machines in the future.

While this work addresses itself to the practical aspects of two-dimensional pattern learning, there already exists a considerable body of work on theoretical one-dimensional learning - grammatical inference. This shows the potential power of a negative sample, (ie. training indicating what not to do) which has interesting implications for the work presented here. The following reference will serve as an introduction to this sphere :

Feldman J. 'Some Decidability Results on Grammatical Inference and Complexity.' Information and Control Vol 20 pp244-262 1972.

CHAPTER 2

RAM DEVICES AS LEARNING PICTURE PROCESSORS

2.1 The General RAM Logic Device

It has long been known that practical pattern recognition machines must incorporate RAM nets. These can form the storage elements within machines utilising techniques such as n-tuples (13), nearest neighbours (19), feature extraction (41) and template matching (37) for pattern recognition. Machines can also be constructed entirely of RAM nets - possibly of many layers - in a closer analogy to a living neural net. These randomly interconnected devices can form nets with considerable information processing abilities. This type of machine is used in the multi-layer net (MLN) approach to pattern recognition (21).

The use of RAM nets in various configurations for pattern recognition comes from the fact that RAMs are universal logic devices. This is a development of earlier work on universal logic circuits using RAMs or RAM-like devices in single layer net (SLN) structures (2).

To illustrate the generality of RAMs, consider any combinational logic block with n binary inputs and m binary outputs. For any given n-bit input word, the block generates an m-bit output word. Now consider a RAM element with an n-bit addressing input and an m-bit data output. This too can generate any combination of m-bit output words when fed with n-bit input words. Evidently the RAM has to be

programmed with the correct data initially to perform this emulation properly. However, it can still in principle perform any of the $(2^m)^{2^n}$ functions that an n-bit input logic device can exhibit at its m-bit output. Hence a RAM loaded with the correct data can emulate any other combinational logic block.

While this has been shown to be possible in theory, the large value of $2^n.m$ - the number of storage bits required in the RAM block - would be impossibly large for a machine of any useful size (4). In addition, the time taken to program it would be inordinately long. However, techniques can be applied to enable practically-sized RAM nets to be used successfully in pattern recognition. These techniques usually involve some form of reduction of the input data resulting in realisable sizes of machine.

2.2 RAM Nets in Picture Processing and Pattern Recognition

The reduction of input data necessary for practical pattern recognition machines usually involves the extraction of subsets or functions from the input data (25,13). Common examples of these subsets are n-tuples and local or global features, which may possibly be combined into feature vectors. These machines are internally composed of identical sections arranged in parallel - one for each class of pattern to be identified. This results in the size of a pattern recognition machine being directly proportional to the number of classes it is capable of resolving. The input pattern passes through all these parallel channels simultaneously and a decision is made as to the most appropriate class label for the pattern.

A process closely allied to pattern recognition is picture processing, where a picture (possibly destined for eventual classification) is transformed into an improved, extended or otherwise modified version of itself. The purpose of this is, typically, to facilitate later processing the picture may undergo - whether by man or machine. However, it is important to note that in the present context when comparing picture processing with pattern recognition, a processor is a single channel device, whereas a classifier is a multi-channel device.

Thus, it may be predicted that such a machine would be smaller in general than an equivalent classifier designed to handle the same type of pictures. A picture processing RAM machine would be of a size comparable with a single class channel of a classifier. This will later be shown to be the case in practice (see Section 2.7).

2.3 The Need to Train RAM Machines

As mentioned in Section 2.1, a RAM machine can emulate any combinational logical device, provided it is programmed with the correct data initially. This pre-programming or 'training' phase is vitally important, as the performance of the RAM machine - whatever its task - depends heavily on the quality and quantity of training received.

For any RAM machine to perform usefully it must generate some dependant output variable that is a definite function of an independent input variable. In training, the machine is presented simultaneously with examples of the input variable (A_{tr}) and the corresponding output variable (B) - generated by the function (f) to be learnt

(ie $B=f(A_{tr})$). In this manner, the RAM machine will learn the function it is to perform on any input variable presented to it in future.

Thus, if the training is adequate the RAM should be able to show coherent performance in a test period. Here, it will be presented with only a new input variable (A_{te}), enabled to read, and expected to generate the new output variable (C, where $C=f(A_{te})$, f being the function learnt earlier).

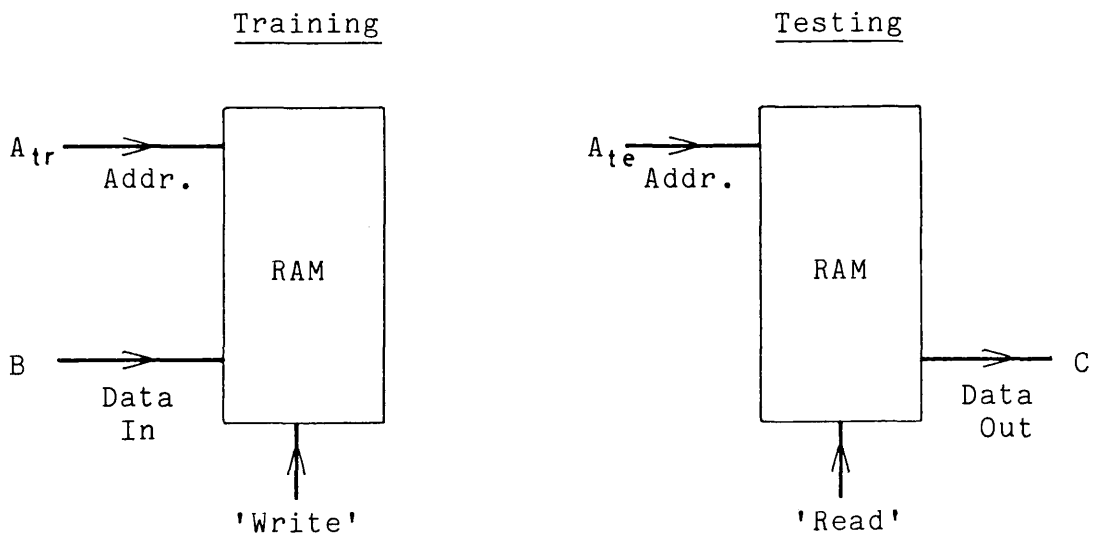


Fig 2.1 Training and Testing Phases of a Learning RAM Device

The usual meanings of the data types A_{tr}, A_{te}, B and C are given below in Fig 2.2 for pattern recognition: this figure also shows a possible set of interpretations for the case of image processing.

Process	Training	Testing
Pattern Recognition	A_{tr} I/P Input Picture (Example of Classification Given Below) B I/P Example of Class Label of Above	A_{te} I/P Input Picture (To be Classified by Machine) C O/P Class Label of Above, Generated By Machine
Picture Processing	A_{tr} I/P Input Picture (Example of Processing Given Below) B I/P Example of Processed Version of Above	A_{te} I/P Input Picture (To be Processed by Machine) C O/P Processed Version of Above, Generated by Machine.

Fig 2.2 Meanings of Inputs and Outputs for RAM Pattern Recognition and Picture Processing Machines

2.4 The Use of a Scanning Window to Reduce Data Input

There is a need in picture processing, as in pattern recognition, to reduce the amount of storage required from that used by the 'brute-force' method shown above. For pictures comprising n bits of data, the storage requirement would be $2^n.n$ bits for such a brute-force picture processing machine. Using practically sized pictures (n=256 or more) this value reaches 10^{78} bits of storage - an obviously impractical value.

The methods used for storage reduction in pattern classification are not necessarily the best or even suitable for doing the same job in picture processing. This is due to the fundamentally different natures of the tasks to be learnt. The former relies primarily on global features and

is not concerned with the exact spatial relationship between adjacent points in the input picture.

(This fact is to some extent justifiable, since if a pattern recognition method is to remove the maximum amount of redundant information from a picture in one major process, in order to classify it, it must retain features that are as nearly as possible uncorrelated. Hence pattern recognition tends to be less concerned with what is happening in local neighbourhoods of a picture.)

Picture processing however, depends on these local features and short range patterns in the input picture. This equal dependence on local patterns - wherever they originate in the picture space - would suggest the use of a small scanning window as a suitable method of reducing the data derived from the input picture.

The application of a scanning window extractor to an n-bit input picture results in the generation of a w_i-bit window for each stage of the scan. This value (w_i) is much less than the number of bits contained in the whole input picture (n); hence a large reduction in the storage requirements can be expected. (It has been assumed in this early discussion that each picture element or pixel is composed of a single binary digit to simplify the analysis.)

To cover the entire picture, this window extractor requires two co-ordinate parameters (x,y) as input to define the current position of the window to be extracted. One complete run will involve the extraction of n separate windows while cycling through all possible values of (x,y) once (Fig 2.3).

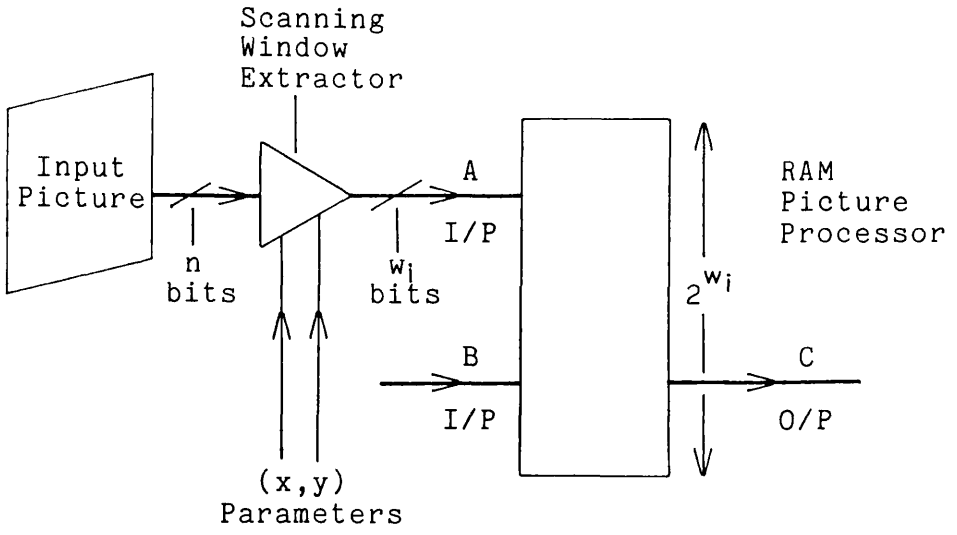


Fig 2.3 Application of a Scanning Window Extractor To the Input Picture

This reduction in the number of inputs at A considerably reduces the storage requirements - by a factor $(2^{w_i}/2^n)$, which is very small. (It should be recalled that $w_i<n$ thus $2^{w_i}<<2^n$.)

The use of a scanning window on the input picture suggests further opportunities to reduce storage, and also to reduce irrelevant data in the example picture at input B. This irrelevant data is the part of the example picture that is not inside the window being currently extracted from the input picture. This stems from the earlier realisation of the fact that a picture processor deals only with local operators, so that only the region of the example picture near (x,y) is of any use in determining the processing operation to be performed on the region near (x,y) in the input picture (Fig 2.4).

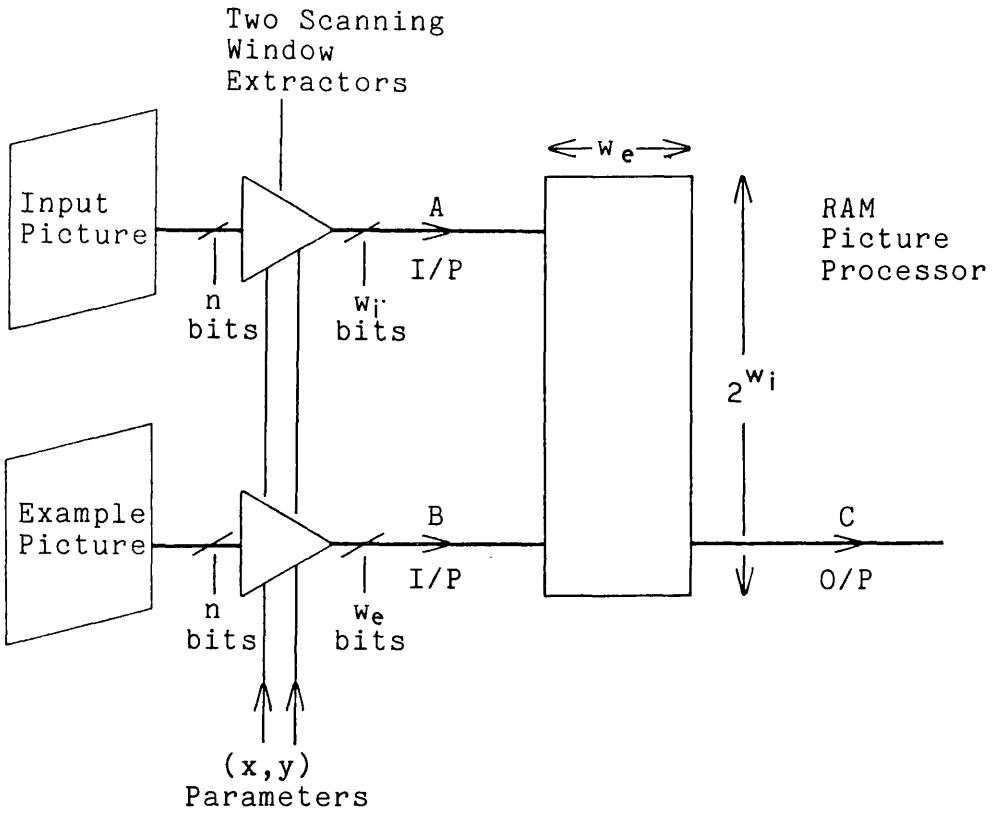


Fig 2.4 The Use of Two Synchronised Scanning Window Extractors on the Input and Example Pictures

This suggests a method of generating the output picture. In training the machine scans the pair of input pictures sequentially. In testing it can similarly scan the input picture to be processed, while synchronously generating the output picture. This can be seen as re-creating a large (n-bit) output picture from the smaller (w_o-bit) data output derived from the RAM memory matrix.

The machine described above uses three synchronized scanning window devices - although only two (shown as I and II below) are used in training, and two (shown as I and III below) are used in testing (Fig 2.5):

Device	Type	Function
I -	(Window)	sequentially extracts
	( )	w_i- (or w_e-) bit windows from
II -	(Extractors)	an n-bit input (or example) picture
III -	(Picture)	sequentially generates
	( )	an n-bit output picture from
	(Generator)	successive w_o-bit windows

Device I acts on the Input Picture in training and testing;

Device II acts on the Example Picture in training only;

Device III acts to form the Output Picture in testing only.

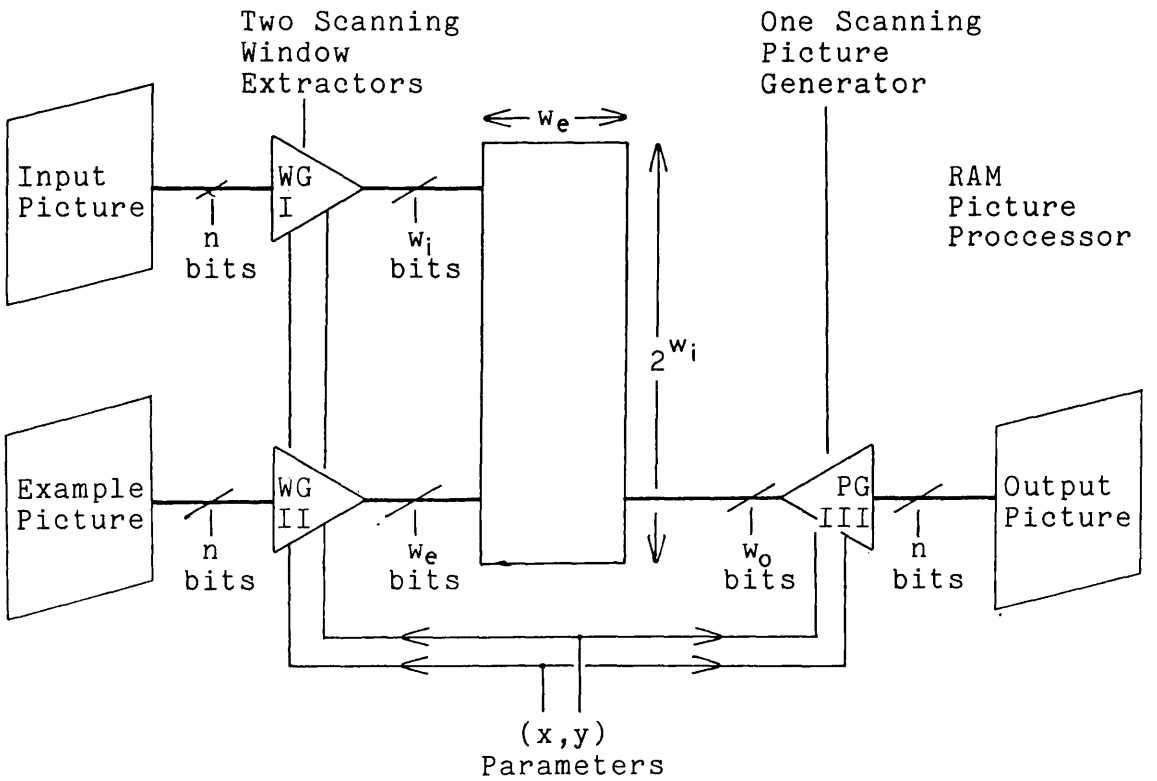


Fig 2.5 The Use of Three Scanning Devices to Read Inputs and Generate Outputs in a RAM Picture Processor

2.5 The Advantages of Using a Scanning Window

This use of a scanning device means that not only is the storage requirement much reduced, but also the training

received from a single pair of input pictures A_{tr} and B is much increased. This is because the number of training sets received from each single pair of pictures is increased to n sets of values. Since the machine receives many more training sets of data from each pair of pictures and can effectively intermix such feature subsets to recreate whole patterns, it can begin to generalize. That is, it gains the ability to be able to process pictures in the testing phase not seen previously in the training phase. This is as opposed to the brute-force machine seen earlier (in Section 2.3) where there is no opportunity to generalize over pictures. Here, each n-bit picture would have addressed one unique cell in the memory matrix, chosen from the 2^n cells present. This machine could not process pictures not seen before, as the memory matrix cells corresponding to those as yet unseen pictures would not have been set before. This property of generalisation is one of the fundamental reasons for using such an architecture, as it is an essential requirement for any practical picture processor, just as it is for any practical pattern recognition machine.

2.6 Processing within the RAM Picture Processing Machine

The actual use made of the incoming data by the RAM machine will now be considered. It has been shown (Section 2.4) that this decrease in the RAM storage requirement results from two causes: the reduction in the incoming data width from n to w_i bits in the case of the input picture, and from n to w_e bits for the example picture. This implies the simplest and, as will be shown later, completely practicable arrangement for processing the

incoming data.

The window extracted from the input picture forms the address lines of the RAM, and the window extracted from the example picture in training forms the data input lines of the RAM. Consequently, the data output lines of the RAM will be used to form the window for generating the output picture during testing (Fig 2.6).

This implies that $w_o \leqslant w_e$, and the simplest and again practical arrangement is to have $w_o = w_e$. That is, the output window used to generate the output picture uses the whole of the original example window stored in training.

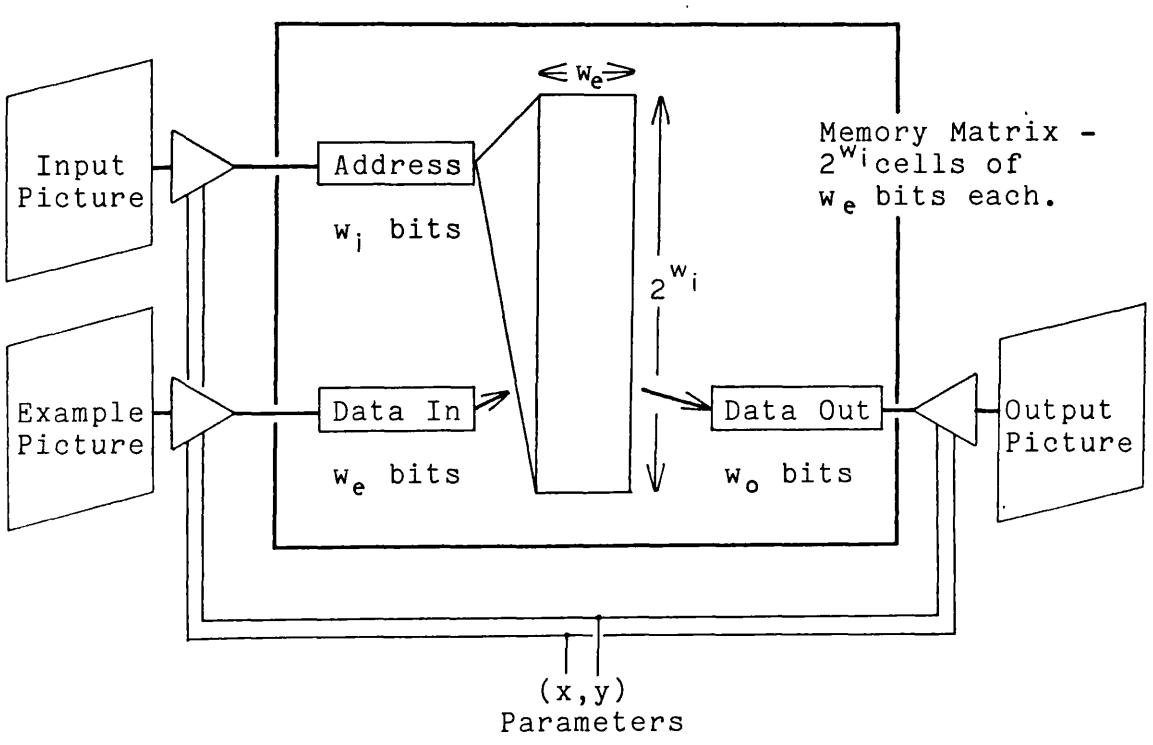


Fig 2.6 Data Handling in a Practical RAM Picture Processing Machine

2.7 Practical RAM Picture Processor Layout

In order to facilitate understanding of the machine in this and the next section, some numerical values for the variables introduced so far are given in Fig 2.7. This will illustrate the layout and operation in actual learning and application of a picture processing task with real data. These numerical values are by no means optimum, but are practical and have been shown to work. (A fuller treatment of the type and range of possible values will be given later in Chapter 4.)

This hardware layout results in the following features:

Window Extractor I

This uses a 256-bit input pattern and a pair of 4-bit (x,y) co-ordinates to define a 9-bit output window. The latter contains the centre point value at (x,y) and its surrounding 8 nearest neighbours. This 9-bit word is used to address one of the 512 ($=2^9$) locations in the memory matrix of the RAM.

Window Extractor II

This similarly takes a 256-bit example pattern and the two 4-bit co-ordinates to define a 1-bit output window, containing only the centre point value at position (x,y). This is used as the data input for the RAM during training.

The RAM

This has 9 address lines, and 1 data input line, giving a store size of 512 1-bit words. The control line defines the mode of operation as being either 'write' or 'read' in training or testing respectively.

Variable	Function	Value	Comment
n	no.of pixels in picture field	256	$(=16^2)$: a coarsely digitized two dimensional picture
w_i	no.of pixels in window extracted from input picture	9	$(=3^2)$: i.e. 3x3 pixel window with a centre point and eight neighbours (implies a rectangular grid)
w_e	no.of pixels in window extracted from example picture	1	the centre point value only, with the same x,y values as above
w_o	no.of pixels in window used to generate output picture	1	a single point is inserted in the output picture at x,y (recall : $w_o=w_e$)
x,y	length of side of digitized picture	16,16	a square picture (n=xy)
b	no.of bits per pixel	1	a binary (black and white) picture

Fig 2.7 Numerical Values of Variables in a Practical LPP Machine

Picture Generator III

This takes a 1-bit data output from the RAM and two 4-bit co-ordinates (x,y) to gradually define a 256-bit output pattern in testing. The pixel value supplied is inserted in the picture field at position (x,y).

(x,y) Co-ordinate Generator

This cyclically scans through all values of x (from 0-15) and for each value of x scans through all values of y (0-15). This covers the entire 256-bit picture

systematically, and 'drives' the above Devices I, II and III.

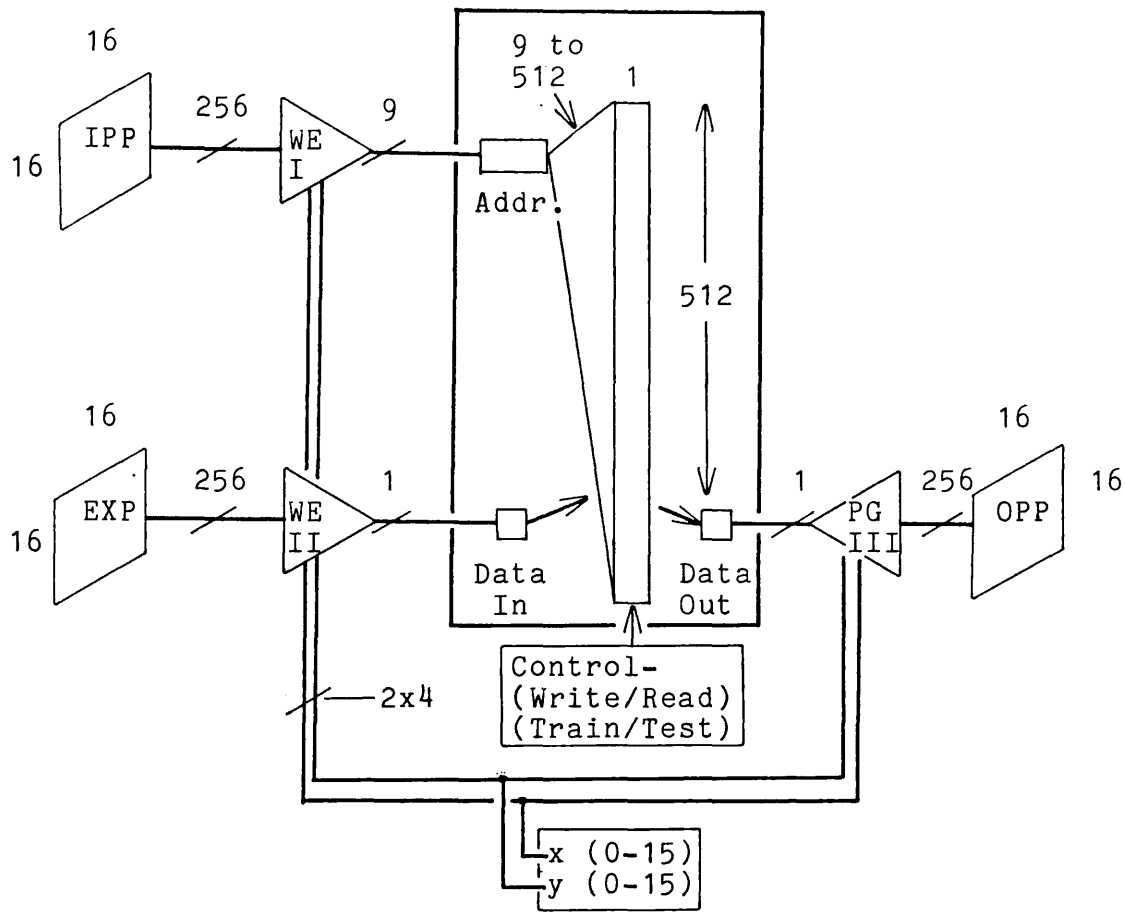


Fig 2.8 Hardware of a Practical RAM Picture Processor

2.8 Mode of Operation

The operation of the machine described above will be described in two phases of operation, training and testing.

Training

The machine is presented simultaneously with two 256-bit pictures - the input pattern (IPP) and the example pattern (EXP). (The example pattern should be a transform of the input pattern, modified according to the processing task

the machine is to learn.) From each point in the input picture the machine extracts a (3x3) window. This is used to form the address, defining one of 512 locations in a 512 by 1-bit RAM array. A single data bit point is simultaneously extracted from the corresponding position in the example picture. This value is written into the RAM location now selected. This process is repeated for each pixel in the pattern field, and then possibly for further pairs of pictures, if the training is to involve more than one pair of pictures.

Testing

The machine is presented with a single 256-bit picture, the input pattern (IPP), and is expected to produce one 256-bit picture - the output pattern (OPP). (This output pattern should ideally be a correctly transformed version of the input pattern, modified according to the process learnt earlier.) For each point in the picture, the machine again extracts a (3x3) window from the input pattern, and uses this to address one of the 512 locations in the 512 by 1-bit RAM array. A 1-bit word is read from this location and is inserted into the picture space of the output pattern at a position corresponding to the current position of the window. The output pattern is correspondingly built up as all points are scanned sequentially.

As stated earlier, it is probable that there will be more than one pair of training examples presented to the machine in the training phase. It is also probable that in the testing phase more than one picture will be presented to the machine for processing, which will in turn be expected to generate a set of output pictures. Thus, in both the

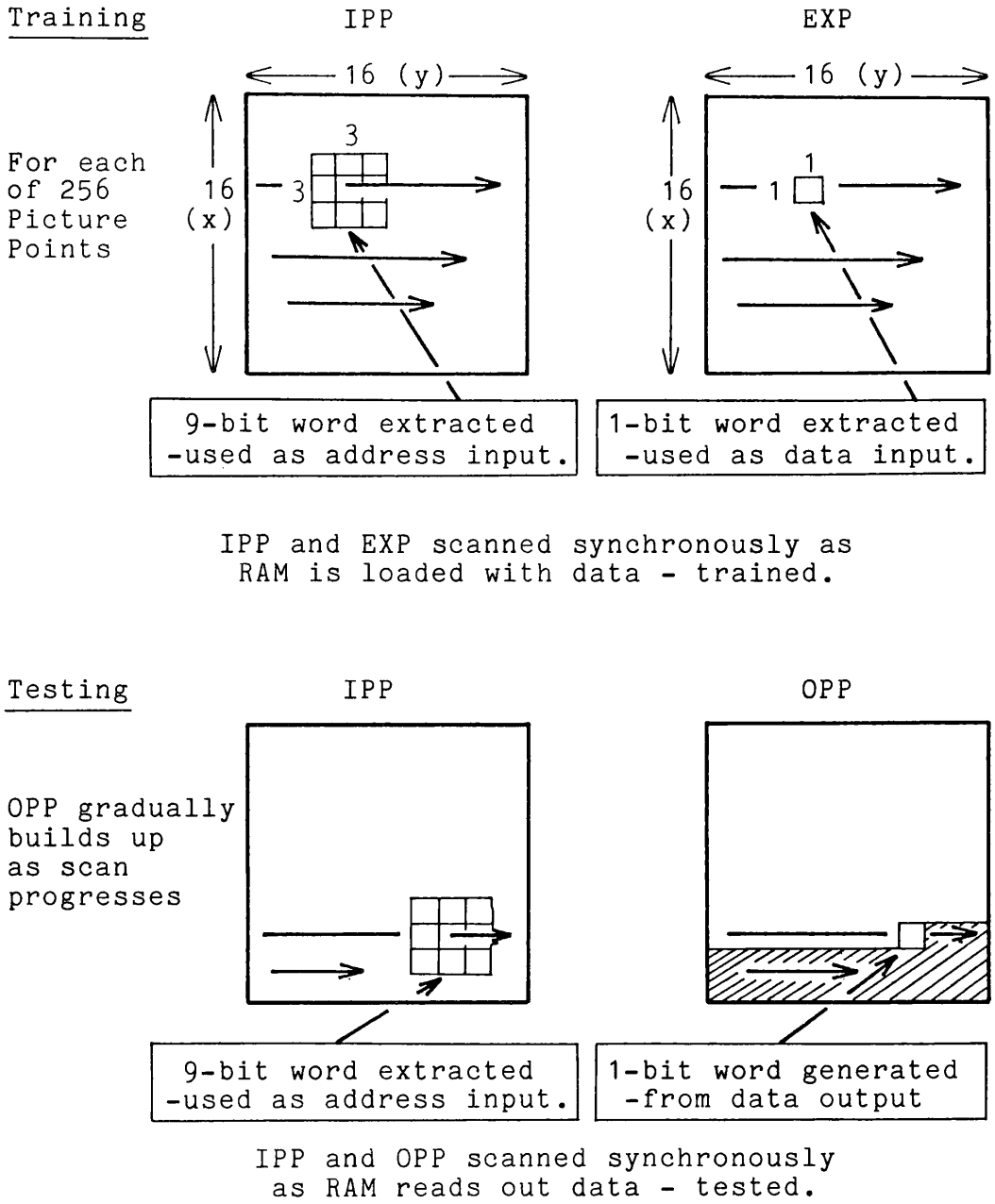


Fig 2.9 Mode of Operation on Picture Pairs

training and testing phases, the machine will be presented with a set of pictures, each set being operated upon as in Fig 2.9. This is shown in Fig 2.10.

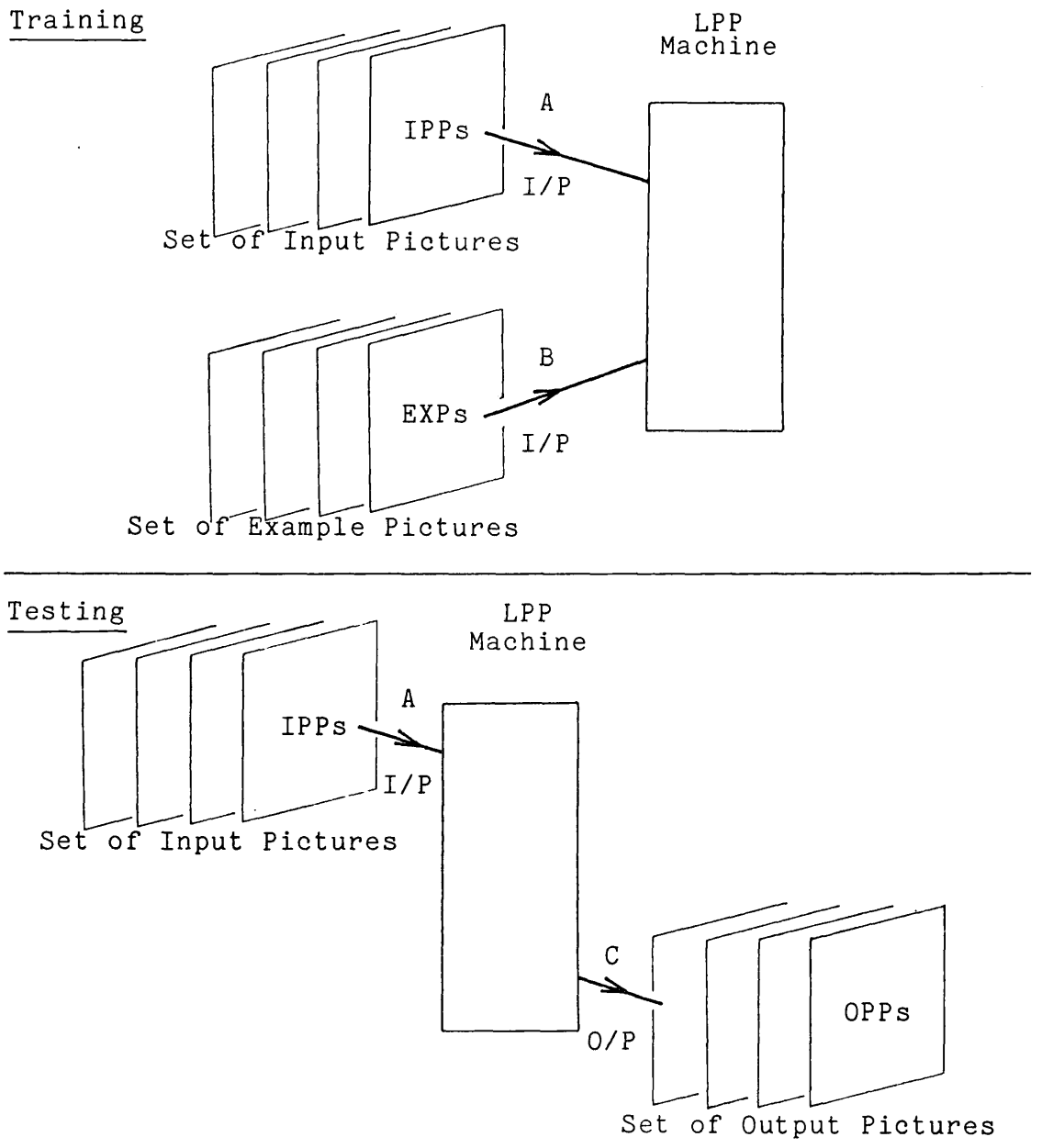


Fig 2.10 Mode of Operation on Sets of Pictures

2.9 Preliminary Conclusions about RAM Picture Processors

This RAM learning picture processor can be interpreted as a simple look-up table of all possible variations in a 9-bit binary window. This is a perfectly valid interpretation, although four additional points should be noted:

1 The look-up table can be written into selectively, and hence it can be generated by examples in training. Thus there is no need to determine how to express the picture processing algorithm analytically in terms of the input window. If examples are available to illustrate the process, this is now a sufficient condition to enable machine processing to be carried out.

2 In addition, the machine's memory matrix, once trained by example, will represent the algorithm that has been used to transform the picture. Thus even if an algorithm is not known, use of such a RAM learning picture processor will generate an expression for this algorithm in terms of a window look-up table. Subsequent analysis of this internal state of the machine will give insight into this initially unknown algorithm.

3 The memory matrix of the machine can also be trained directly, as opposed to being trained by example. If the picture processing algorithm is known analytically, then the memory matrix locations can be loaded directly with the correct data. The machine then operating in the testing phase would perform this picture processing algorithm as if it had been learnt earlier by example. This procedure would have the particular advantage of rapid processing of known algorithms (see below).

4 The Learning Picture Processor can in principle operate extremely rapidly, since the only processing time required is the generation delay for a window (usually a simple extraction of a subset of data from the input set), coupled with the access time of the RAM, for each point in the

picture. To take the simple example of the binary picture used earlier with 256 (16^2) picture points and a RAM having an access time of 200ns, then a picture could be processed in approximately 50μs. Picture points are often generated serially (as in, for example, a video system) and can be fed into a shift register type processor, that effectively extracts windows from high speed serial data in a 'shift delay' period. If this delay period is less than or equal to the access time of the RAM, then a processing time of 50μs would become realisable in practice.

CHAPTER 3

EXPERIMENTS WITH BASIC LPP MACHINES

3.1 An Introduction to the Experimental Work

The hardware layout and mode of operation of a small, simple Learning Picture Processor (LPP) have been described in Sections 2.7 and 2.8. It will be appreciated that this layout is neither the only possible nor even an optimum one. It is evident that a very large number of variations is possible. A simple layout was chosen merely to serve as a workable example of an LPP machine. This particular layout will be used in the following experiments to show the concept of machine learning of picture processing to be practicable. Similarly, the mode of operation described is one of the simplest possible, and will serve to show the machine as described performing actual picture processing operations.

Both these aspects can be considerably extended and ultimately improved. These improvements will stem from :

1 theoretical considerations of the processes occurring in the machine, which will suggest layouts and operating modes more efficient in terms of speed, performance, flexibility and quantity of hardware used,

2 results of practical experiments which will confirm or suggest new methods of improving the machine.

As the preliminary experiments are run, improvements will gradually be incorporated into the machine and their results noted. Eventually, a global view of the machine in its most general form will emerge and will be fully discussed in Chapter 4. More ambitious variations in the layout and operation of an LPP machine will then be investigated in the further experiments in Chapters 5 and 6.

3.2 Picture Processing Tasks for the Simple LPP Machine

In order to attempt picture processing with the LPP machine, it is necessary to consider first the range of tasks the machine should be able to tackle. Some physical constraints are apparent with the machine as described in Section 2.7. It is only within these limitations that the machine can be expected to perform usefully. There are essentially two such constraints :

1 The use of a binary digitized picture; ie. only two grey levels are used to define the brightness at each picture point (most conveniently labelled 'black' and 'white'),

2 only the centre point and its immediate eight connected neighbours on a rectangular lattice will be used to define and execute the picture processing algorithm.

These constraints could well be relaxed later on: until then several useful picture processes are still possible.

Examples of such operations are:

Inversion

The swapping of the two brightness values to form a 'negative' image, ie. white objects on a black background. (Here, and in what follows, an initial picture is assumed to be composed of black objects on a white background.)

Cleaning

The removal of small isolated black (or white) specks from a contrasting white (or black) background. This is a well known operation which improves the picture by the removal of 'salt and pepper' noise (66).

Shrinking

The removal of a single outer layer of black points from black objects, often with a view to removing black objects entirely below a certain size.

Expanding

The removal of a single outer layer of white points from white objects, often to remove white holes entirely below a certain size. Shrinking and expanding are sometimes applied consecutively to remove both black and white objects and holes (65).

Shifting

The movement of objects in a given direction with respect to their (stationary) background. This serves as an aid to subsequent position dependent analysis of objects (45).

Edge Finding

The location and marking of points on the boundary between black and white areas resulting in the outline of the objects - a useful aid in classification and scene analysis (69).

Convex Hull Determination

The transformation of black objects into their minimum enclosing convex shapes by the addition of black points into concavities. This resulting convex hull can then be used in certain methods of pattern recognition and shape description (53).

Location of Objects

The identification and marking of objects of a particular shape or size within the field.

Thinning

The production of a unit width black skeleton from a black object with variable width limbs, whilst maintaining limb ends and connectivity. This is a well known aid to pattern analysis (70,10).

This list is clearly not exhaustive: it is limited to include only those picture processing tasks that the LPP machine described earlier could perform. However, it contains examples of varying complexity from such trivially simple tasks as inversion and cleaning, up to more involved tasks such as the location of objects or thinning. (These may require several passes through a small-windowed and thus diameter-limited machine (50), performing the process gradually on successive passes.)

One important point should be mentioned at this stage regarding the concept of a Learning Picture Processor and the tasks it can perform. The above list of picture processing tasks contains a range of well-defined separate operations that may be of use in transforming a picture for subsequent classification or identification. These tasks have already been compartmentalized in that each one has been graded in complexity and does a specific, limited job: some suggestion has even been made of the algorithm that might be used to do this job.

One of the predominant features of a Learning Picture Processor to be demonstrated here is its generation of its own processing algorithms when presented with examples of an original and transformed picture. This transformation need not be simple nor even immediately reducible to any of the picture processing operations listed above. Provision of examples showing the required transformation is sufficient to train the machine, without recourse to direct investigation of that transformation itself.

3.3 An Outline of the Computer LPP Simulator

An LPP machine was first constructed by software simulation of the hardware on a small general purpose digital computer. Since large-scale changes to the ultimate layout were envisaged as the experimental investigation proceeded, simulation had many advantages over actual construction of working hardware. These included development time, effort, flexibility, ease of connection to existing picture handling peripherals and the possibility of

including some form of rudimentary operating system. This facilitated the handling of data and the automation of some repetitive processes.

The software package developed to fulfil these requirements was written in Assembly Language for a Motorola M6800-based microcomputer (49,52). This use of a local machine was aimed at making the maximum use of available image input and output devices. A low level language was used to maximize the machine's effective size and speed. The program proceeds as a series of interactive exchanges between the computer and the operator, taking the form of machine prompts, operator responses and machine processing.

This description of the simulation software has been provided here to facilitate understanding of how the following experiments were performed in practice. A more complete description is given in Chapter 7.

3.4 Experiment 1 : Proving Run

As an initial run using an LPP machine for the first time, a simple picture processing task was attempted. This task was the thinning down of a character with limbs of three to five units width to limbs of two to three units. It should be remembered that no precise definition of the operation is necessary, as the provision of examples is itself sufficient definition.

Data

The original letters used as the input patterns (IPPs) in these early experiments were hand-written block capital letters collected from volunteers and digitized on a binary

16x16 matrix. Details of the collection of these data is given in (58). The example patterns (EXPs) were produced by subsequent manual editing of these pictures to remove noise points and generally thin the limbs down to an approximately constant width by eye. This produced a set of non-rigorously transformed data - suitable for testing a LPP machine's learning ability.

Training

The input pattern (IPP) and example pattern (EXP) presented to the machine are shown in Fig 3.1. The EXP can be seen as a thinned version of the IPP. This one pair of pictures is presented to the machine and the RAM is enabled to write - ie. it is trained. This occurs separately for every pixel, as training occurs in parallel for such a machine.

Testing

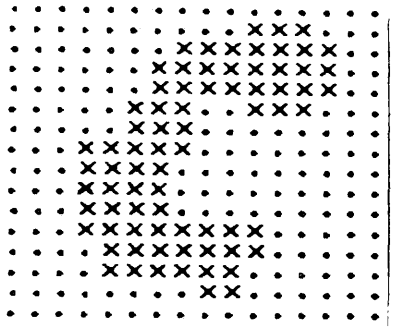
To test the performance, another input pattern (IPP 1) is presented to the machine and the RAM is enabled to read, thus generating an output pattern (OPP 1) as the machine scans through the picture field. This similarly occurs in the parallel mode, as the machine builds up OPP 1 in a different picture space from IPP 1. (Later experiments in Chapter 6 will also use the sequential mode of operation.) This testing IPP 1 is shown together with the resultant OPP 1 in Fig 3.2.

Results

The machine can be seen to have transformed the testing IPP 1 in generally the same manner as the training IPP was transformed - it has reduced the thick object to a thinner

Training Set : 1x(IPP+EXP) pair presented to the machine

IPP



EXP

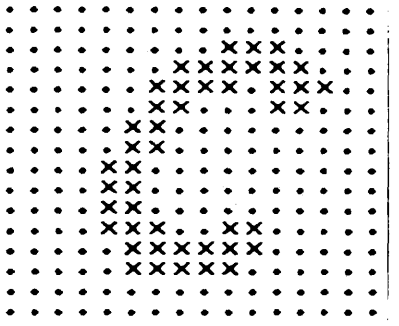
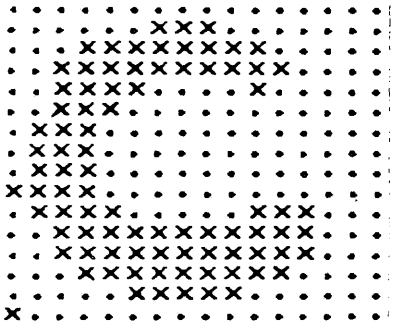


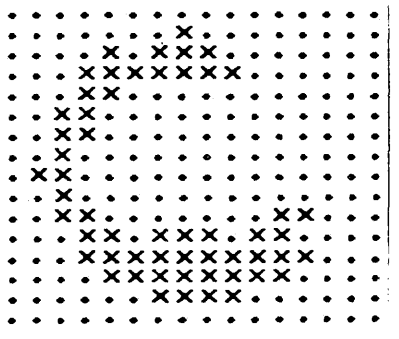
Fig 3.1 Experiment 1 : Training Patterns

Test Set : IPPs 1-2 presented to machine,
which generates OPPs 1-2 respectively

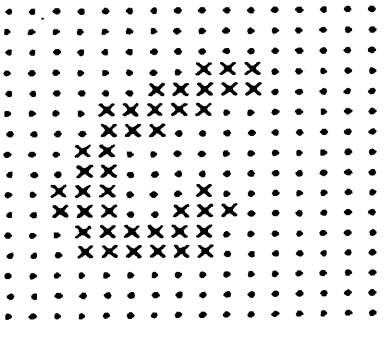
IPP 1



OPP 1



IPP 2



OPP 2

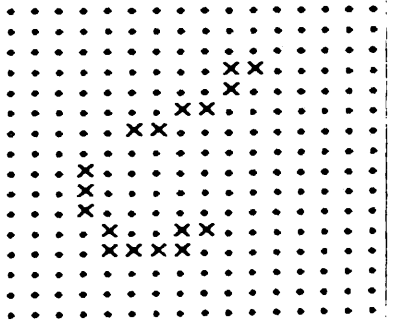


Fig 3.2 Experiment 1 : Testing Patterns

one. This been achieved after the use of only one pair of training examples. The fact that the machine responded correctly after only one such training pair is an important point and will be referred to later.

Discussion

Despite the success in producing a thinning effect as trained, there are some shortcomings in the performance obvious from even a cursory inspection. The OPP 1 now has erratic shape and limb widths when compared with the original. Although the average limb width has decreased, it now ranges from one to four units. This would appear to be due to 'under-training' - ie. not all bits in the memory matrix were actively set or reset by the training patterns. The memory matrix was arbitrarily cleared (all bits set to '0') before training. This means that bits actively cleared to '0' by training are indistinguishable in testing from bits initially cleared to '0' and thereafter not accessed in training. Thus if, in testing, IPP 1 presents a (3x3) window feature not seen before in training, this will address a cell in the memory matrix not accessed since pre-training initialisation. Consequently it will contain (and hence read out) a '0' into this position of OPP 1. These additional '0' points in OPP 1 can thus be attributed to insufficient training. This will be investigated further in Section 3.6, Experiment 3.)

As a further demonstration of the inadequacy of the training received, an already thin pattern was also used to test the same machine. This pattern IPP 2 and the machine's output OPP 2 are shown in Fig 3.2.

The resultant broken limbs again suggest insufficient training. The machine has never seen patterns with features like those in IPP 2 before, and consequently cannot be expected to produce correct transformations of such patterns. This is exactly analogous to the well-known RAM pattern recognition machine that might exhibit poor classification ability due to insufficient or unrepresentative training. This results in too few discriminators calling a co-incidence with the unfamiliar pattern and hence an unreliable (and probably wrong) result occurs. Attempts have been made to avoid this in established recognition systems, involving choices of 'optimum' internal arrangements (12).

In the case of the LPP machine, the wrong result takes the form of the introduction of excessive '0' values into the output picture space, here totally breaking the original object.

3.5 Experiment 2 : Training Set Size

In an effort to improve the performance of the LPP machine, it has been noted that it requires sufficient examples in training to enable it to generalize over a greater range of pictures in testing. This can be effected in two ways :

1 The training patterns themselves can be altered to contain more varied features, in an attempt to train more memory matrix cells per pattern. Every training pattern of n pixels each effectively constitutes n training patterns, as n w_i-bit windows are extracted from each one.

This explains the apparently reasonable picture processing ability found after just one pair of training patterns in the previous experiment. The single training pair generated n (256) training patterns on scanning the input picture, pausing at each of the 256 points. So, if as many different features as possible (of all 2^9 possible window features) can be introduced into a single training pair, then that pair will train the machine to a far greater extent than one with less variation. (This approach to increasing the training will be attempted in Experiment 13.)

2 More training patterns can be used, in the hope that more patterns will be likely to contain a wider variation of features. This simple and obvious method of increasing the amount of training will be tried here.

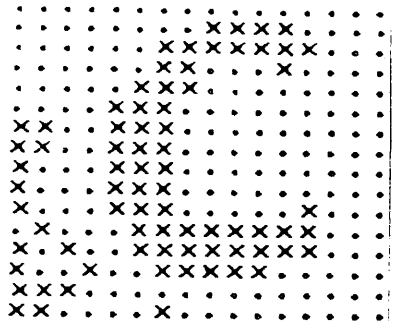
Training

A set of eight pairs of thick patterns (IPPs 1-8) and corresponding thin patterns (EXPs 1-8) is presented to the machine for training. These are shown in Figs 3.3 and 3.4. The memory matrix of the RAM is initially cleared before training such that all bits are reset to '0'.

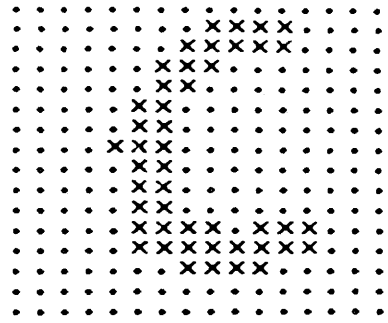
Testing

Again, the parallel mode of operation is used to generate the OPP set. The test IPPs are shown together with their corresponding OPPs in Fig 3.5; as are the two outputs generated by the machine in Experiment 1 - to facilitate comparison between these two machines with differing amounts of training. This test set (IPPs 1-4) contains the original two characters used to test the machine in Experiment 1 (a thick and thin letter 'C') and also two letters of a

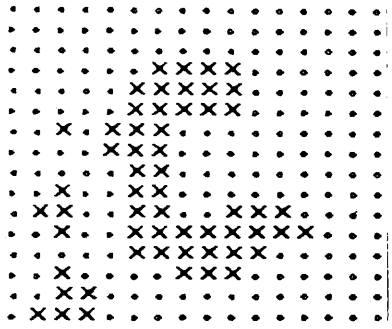
IPP 1



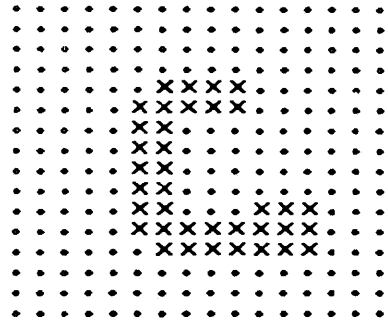
EXP 1



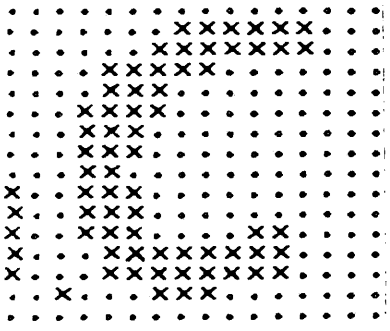
IPP 2



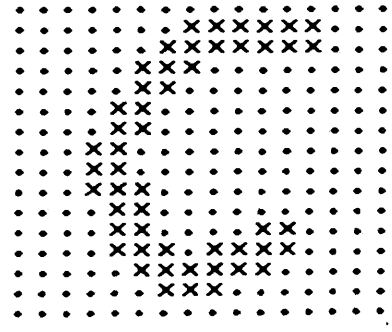
EXP 2



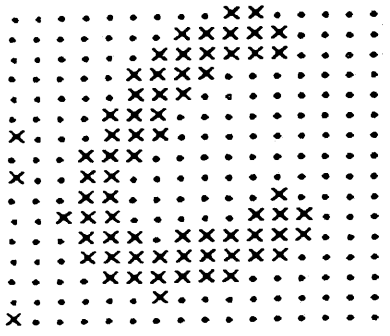
IPP 3



EXP 3



IPP 4



EXP 4

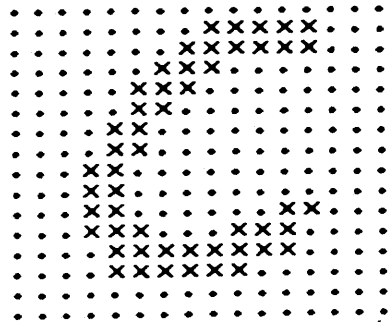
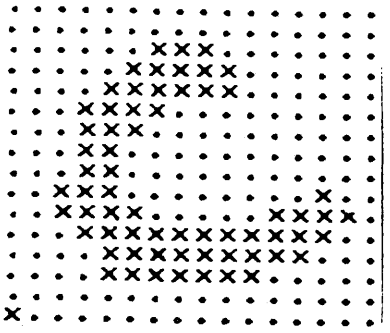
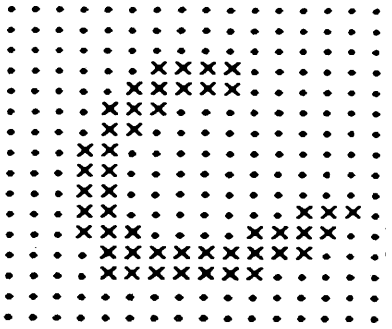


Fig 3.3 Experiment 2 : Training Patterns 1-4

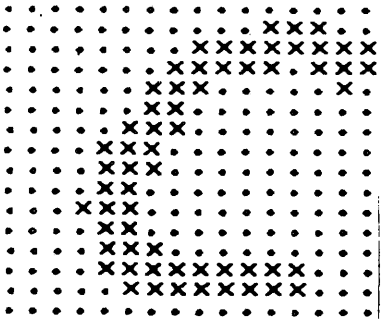
IPP 5



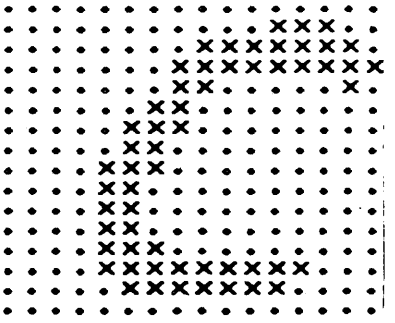
EXP 5



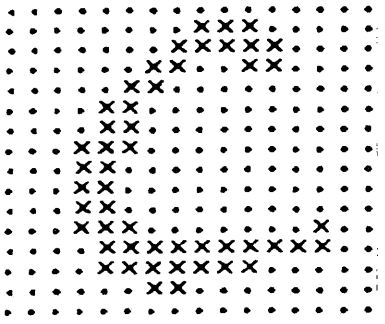
IPP 6



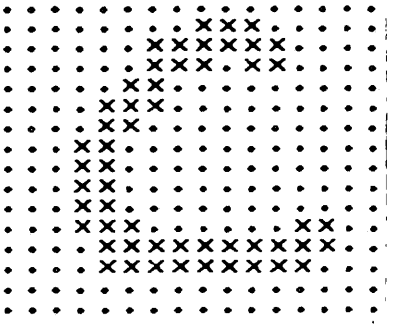
EXP 6



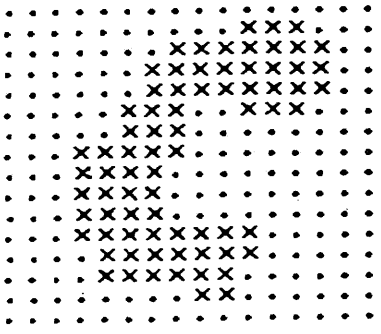
IPP 7



EXP 7



IPP 8



EXP 8

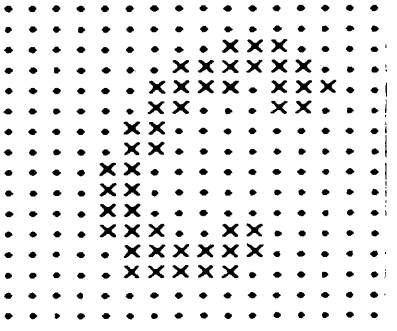


Fig 3.4 Experiment 2 : Training Patterns 5-8

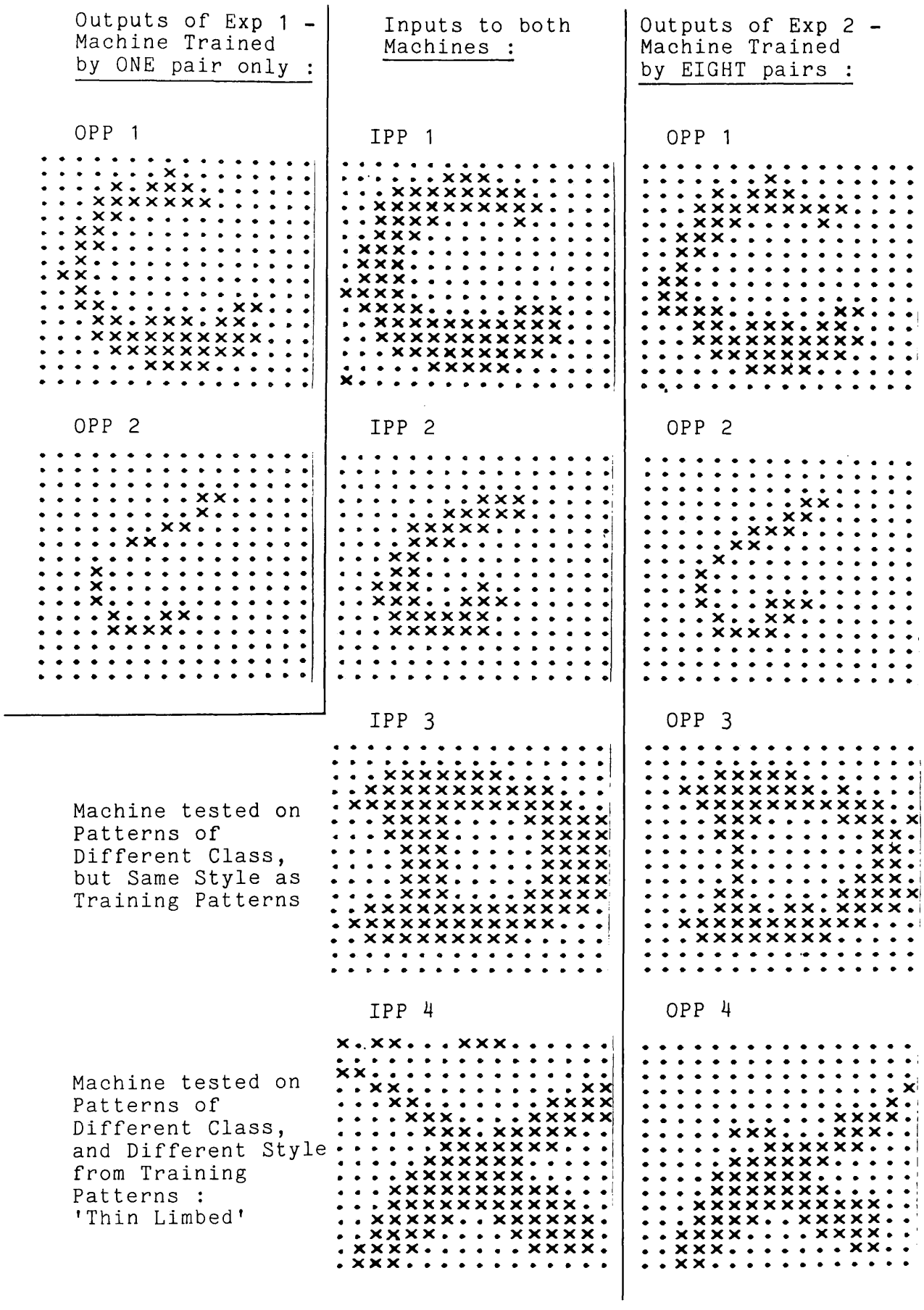


Fig 3.5 Experiments 1 and 2 : Comparison of Test Results

different alphabetic class - the letters 'D' and 'X'.

Results

The OPPs 1-2 (Fig 3.5) generated by this machine (in the right hand column of the diagram) are to be compared with the OPPs 1-2 generated by the same machine in Experiment 1 (left hand column of diagram). In both these cases the same IPPs 1-2 were used (centre column), the only variation being the quantity of training received.

Discussion

It can be seen that in the case of IPP 1 both machines produced a thinner version of the object. This is to be expected, since both machines are now being tested with the same type of data (ie. thick letter 'C's) as those on which they were trained. However, when the second test patterns are examined (IPP 2 to OPP 2) for both these machines, a thinner object being presented as a test, the two results begin to show the differences in training. The machine trained on only one pair has broken the limbs completely at one point, but the machine trained on eight pairs has retained connectedness throughout the object, producing a thinned, but not broken skeleton.

As a further test of this machine's ability, it was tested with an object of a different class (IPP 3) : the letter 'D'. The resultant output (OPP 3) shows that the machine is capable of processing objects of a class different from that on which it was trained. In hindsight, this is to be expected since the machine as described relies only on local features as a small (3x3) window is used. Since there is no dependence on similarity between global

features in the training and test set the machine is 'blind' to any large scale shapes of objects: it is essentially 'diameter-limited' (50).

A final test was made to find the limit of useful performance of this machine. An object (IPP 4 : the letter 'X') containing limbs of varying thickness, from one to five units width, was presented to the machine. This and the resultant OPP 4 are shown in Fig 3.5. The machine removed the narrow limb entirely, illustrating that although it has been more thoroughly trained than that of Experiment 1, it is still deficient. In fact there are still some memory matrix cells unset, as not all the possible 512 ($=2^9$) window features have been seen in training.

3.6 Experiment 3 : Initialisation

The fact that the training given to the LPP machines was insufficient in the above two cases has been stated without proof. To furnish proof, the following experiment was performed.

The LPP machine was set up exactly as for Experiments 1 and 2, except that the memory matrix cells were all initialised before training commenced to contain the value '1' rather than '0'.

Training

The training given was exactly the same as that in Experiment 2 so that a comparison of the ultimate results can be drawn.

Testing

For ease of comparison with the results of Experiment 2, the same four picture objects are used again here. These IPPs 1-4 are reproduced in the centre column of Fig 3.6, with the corresponding OPPs 1-4 in the right hand column. (The output of Experiment 2 (OPPs 1-4) is reproduced in the left hand column of the same diagram.)

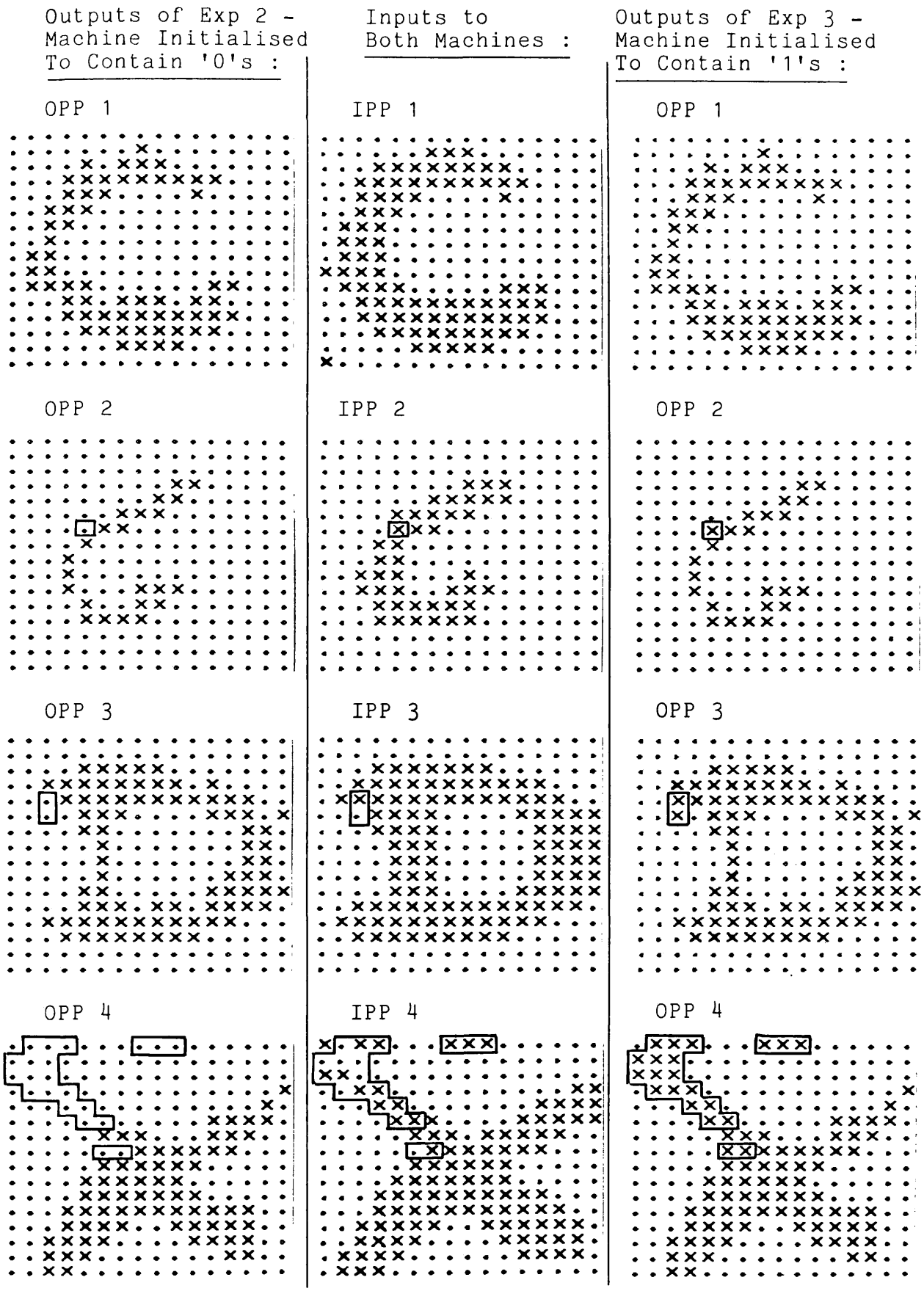
The only difference between this and the previous experiment lies in the initial states of the machines - training and testing being identical. This means that any difference in the final outputs must be due solely to the accessing of cells in testing that were not accessed at all in training, and hence would be still in their (different) initial states. The position and extent of these differences will indicate the cells in the memory matrix not trained by the training set.

Results

An examination of Fig 3.6 reveals the expected differences in the OPPs of the two Experiments 2 and 3. The differing pixels have been 'boxed' in the diagram.

Discussion

If these boxed regions are examined on the test IPPs (centre column of Fig 3.6) they show that the very regions where the OPPs differ is where the IPPs contain the type of features not seen before in training. This leads to the conclusion that the chosen training set is inadequate in the sense that it has now been shown to have insufficient variation (ie. range of features) to enable the machine to generalize correctly. At least, the machine can generalize



Differences in OPPs are 'boxed' on corresponding patterns

Fig 3.6 Experiments 2 and 3 : Comparison of Test Results

to a limited extent - to include IPPs 1-3 which are broadly similar to the original training set, but not IPP 4 which contains features (viz. thin limbs) not found at all in the training set.

Thus, it can be seen that while increasing the training set size demonstrably increases the quantity of training received, the resultant machines could still be even more fully trained. Experiment 3 has also shown how the machine's performance is heavily dependent on the initialisation of the memory matrix before training. To be specific, if the machine is not fully trained (in that the training set does not contain all possible features) then it will generate its initialisation value when attempting to process an unseen feature. This can have a considerable effect of the overall appearance of an output when the test input varies greatly from the training inputs.

3.7 Variations in the Memory Matrix Format

The previous experiments have shown how the performance of the machine depends on a change in the initialisation of the two-state memory cells. This raises the question of which of the many possible internal arrangements of such a machine is optimal.

A brief description of the format used in the experiments performed so far will be given as an aid to the suggestion of variations on this arrangement.

The LPP machine used so far has consisted of a set of bistable memory cells which are addressed by the contents of binary windows extracted from the input picture IPP. In

training, the memory matrix cells thus addressed assume the binary values in the corresponding centre point positions in the example picture EXP. This occurs for every point in the picture field. In testing, windows from a new IPP address memory matrix cells whose contents are placed in the output field OPP in the positions corresponding to the current IPP window.

These memory matrix cells have only two possible states and so once a cell has been set, there can be no further reaction by that cell to any further setting stimuli received in training thereafter. Similarly, a cleared cell cannot respond to further clearing stimuli. The cells may oscillate if the stimuli received set and clear the cells alternately, but this condition only shows inconsistent training. Repeated stimuli in the same sense contain valuable information that a common feature has been found. This information is effectively being lost.

A better LPP machine might well record this as the number of sightings of each particular feature. This would require the storage of a multi-level variable in the memory matrix, as opposed to the present two-level system.

In <u>training</u>, the binary centre point value extracted from EXP could be used to cause a fixed small increase (if this point is '1') or decrease (if '0') in the current value of the memory matrix cell being addressed by the IPP window. These cells can be initialised to contain some intermediate value, and the increment or decrement could be made small compared to the maximum range of the values in the cells. The machine should then be able to record relatively accurately the number of times each cell has been accessed,

at the same time taking note of the polarity of the training stimulus.

In testing, the cells addressed by the windows extracted from IPP now effectively contain analogue variables. These are high if a large number of positive stimuli (EXP point ='1') were received or low if a larger number of negative stimuli (EXP point ='0') were received. A simple process to produce the required binary output would compare the cell content with a threshold value and output a '0' pixel (into OPP) if the contents are lower than this threshold or a '1' pixel otherwise.

A summary of this new internal arrangement (which will be labelled 'Format 2' for future reference) is compared with the arrangement used earlier ('Format 1') in Fig 3.7. This new Format 2 introduces four more parameters that must be chosen before a working system is simulated :

1 The new size of the memory matrix cell. As an eight-bit computer was used for the simulation and eight bits give a range of 256 levels, this was deemed a suitable starting point for the investigations.

2 The effect of the EXP centre point value in training. The increment or decrement chosen by the polarity of the training stimulus causes a change in the cell contents of one level. This is the obvious choice in this case.

3 The initialisation value of the cells. So that each cell should be able to record a change in both directions an intermediate value ('127') was chosen. This would stop cells from reaching either the maximum ('255') or minimum ('0') levels prematurely. (If these levels are reached, the

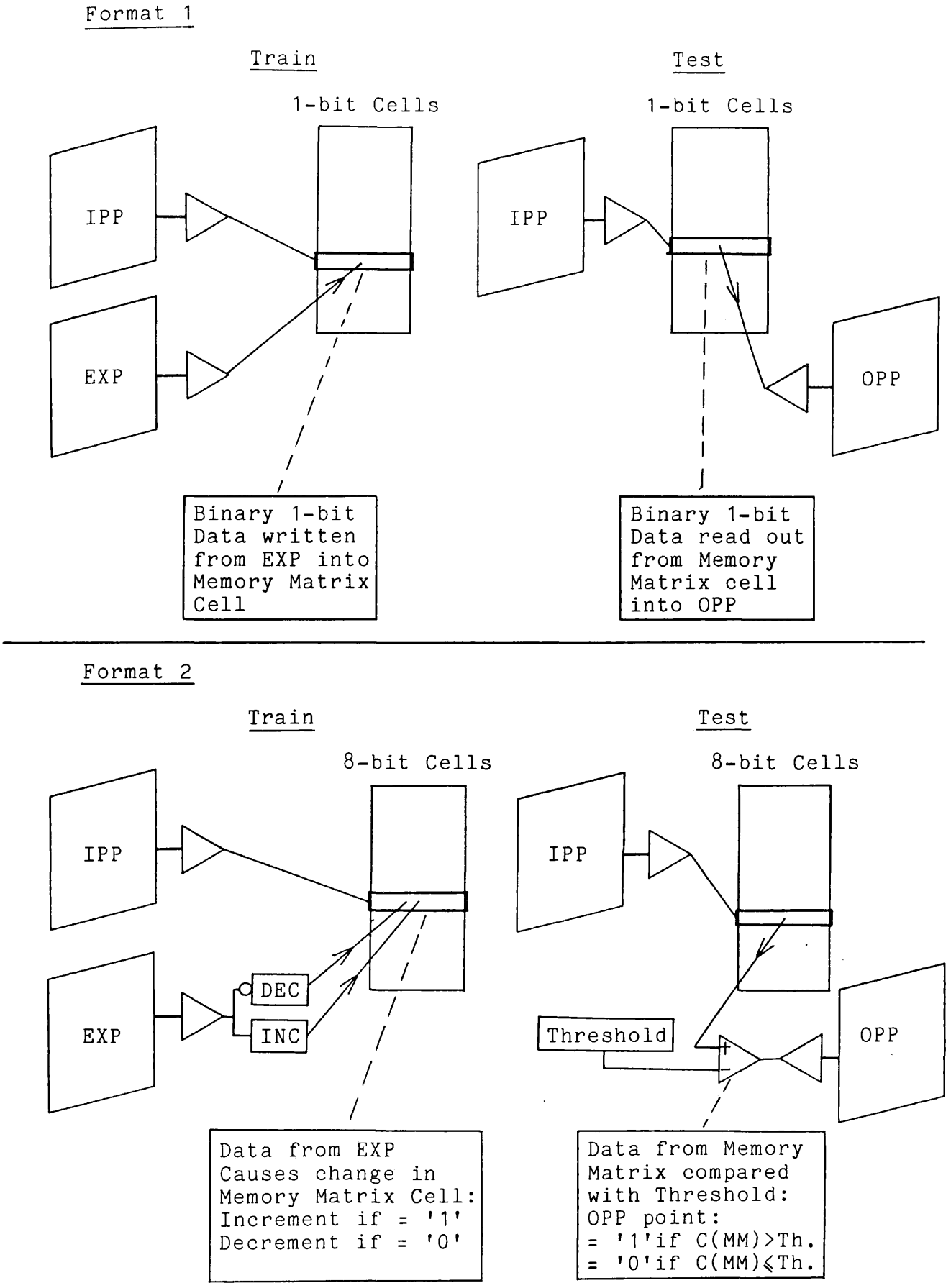


Fig 3.7 Comparison of Internal LPP Formats 1 and 2

machine would not respond to further stimuli in the direction of the limiting.)

4 The threshold level at which, in testing, a choice is made of the binary OPP value to be generated. This was originally set equal to the initial value of the cells, such that any stimuli in either direction, however small, would show up as a resultant change in output.

3.8 Experiment 4 : Two and Many Valued M.M. Cells

This experiment will investigate the relative performances of the two types of LPP machine formats now introduced :

Format 1 (hereinafter referred to as 'F1')

where the memory matrix cells have just two levels, and the cells are set to equal the binary training stimuli received from EXP in training; and are output directly to generate OPP in testing,

Format 2 ('F2')

where the memory matrix cells have 256 levels and are incremented or decremented by one level according to the polarity of each binary stimulus received from EXP in training; and are 'thresholded' to produce a binary output to generate OPP in testing.

Training

To ensure that neither the F1 nor F2 LPP machines is undertrained, a large training set size of 200 pairs was used. This should ensure that the majority of the cells in

the F1 machine each receive some training, such that few are left in their 'initialisation' state which can have a large effect on the results (see Experiment 3). This should also allow cells in the F2 machine to assume values widely distributed in the range of '0-255' levels.

Both machines received the same training set of 200 pairs of IPP and EXP patterns - again drawn from the stock of manually thinned and cleaned characters, similar to those in Fig 3.3. The exact character set used contained 100 pairs of the letter 'C' and 100 pairs of the letter 'D' and are not reproduced here. As each pair effectively generates 256 actual training pairs, it is assumed that the resultant 51200 (=256x200) sub-patterns would fully train the 512 cells in both memory matrices. The danger of 'overtraining' - a common problem in RAM pattern recognition nets - should not arise here. This is because unlike many pattern recognition systems, these cells can repeatedly be set and cleared (F1) or adjusted (F2) to reflect ever more accurately the training stimuli received. The nature of the problem is also different: in picture processing an attempt is being made to maximise the response, whereas in pattern recognition a differential response between discriminators is being sought.

Testing

The same test set (IPPs 1-12) was used to test both machines, and is shown in the centre columns of Figs 3.8 to 3.10. The OPPs 1-12 generated by the F1 machine are shown in the left hand columns, and the OPPs 1-12 of the F2 machine on the right. A relatively large test set of various character fonts was used in an attempt to show up any

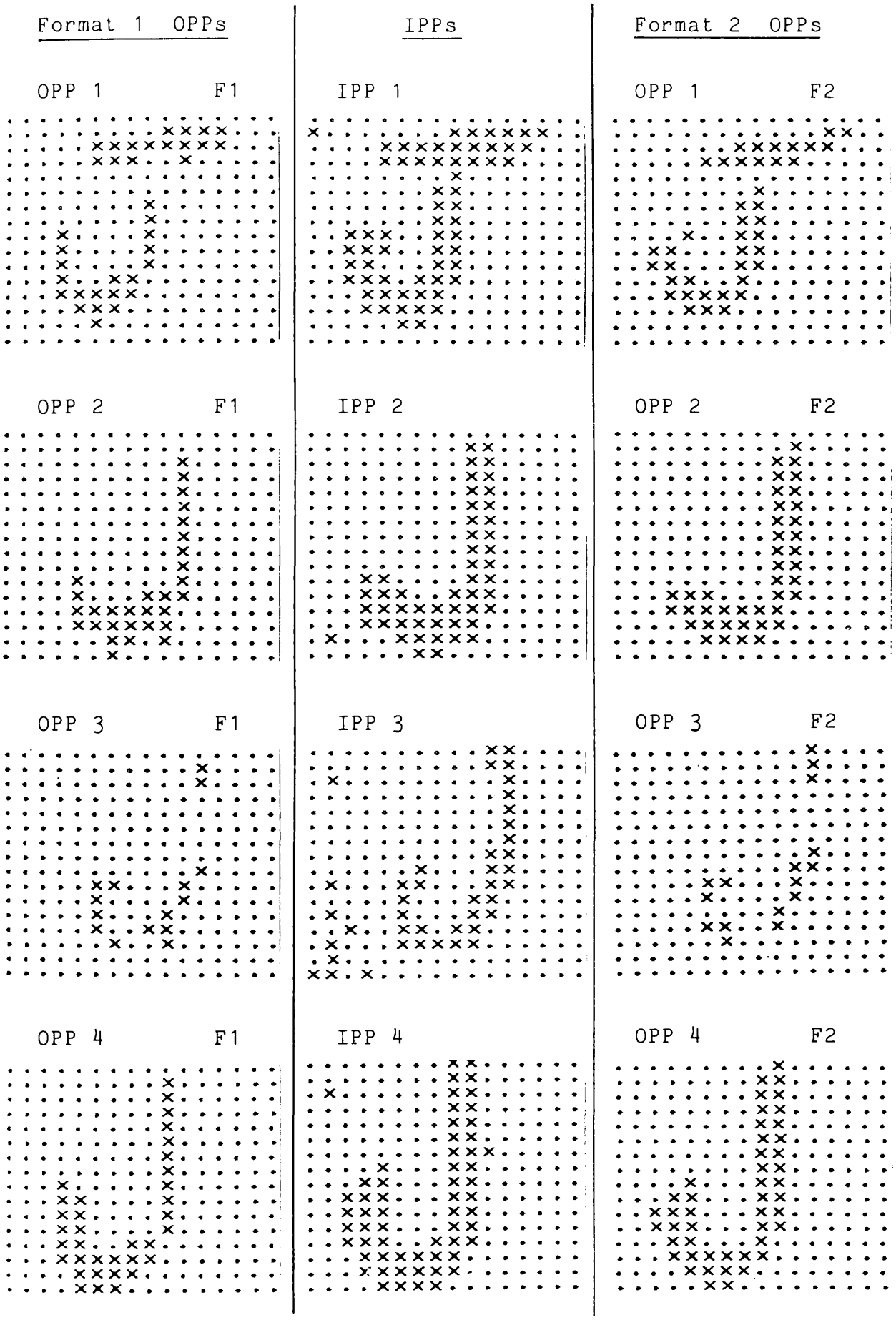


Fig 3.8 Experiment 4 Test Patterns 1-4

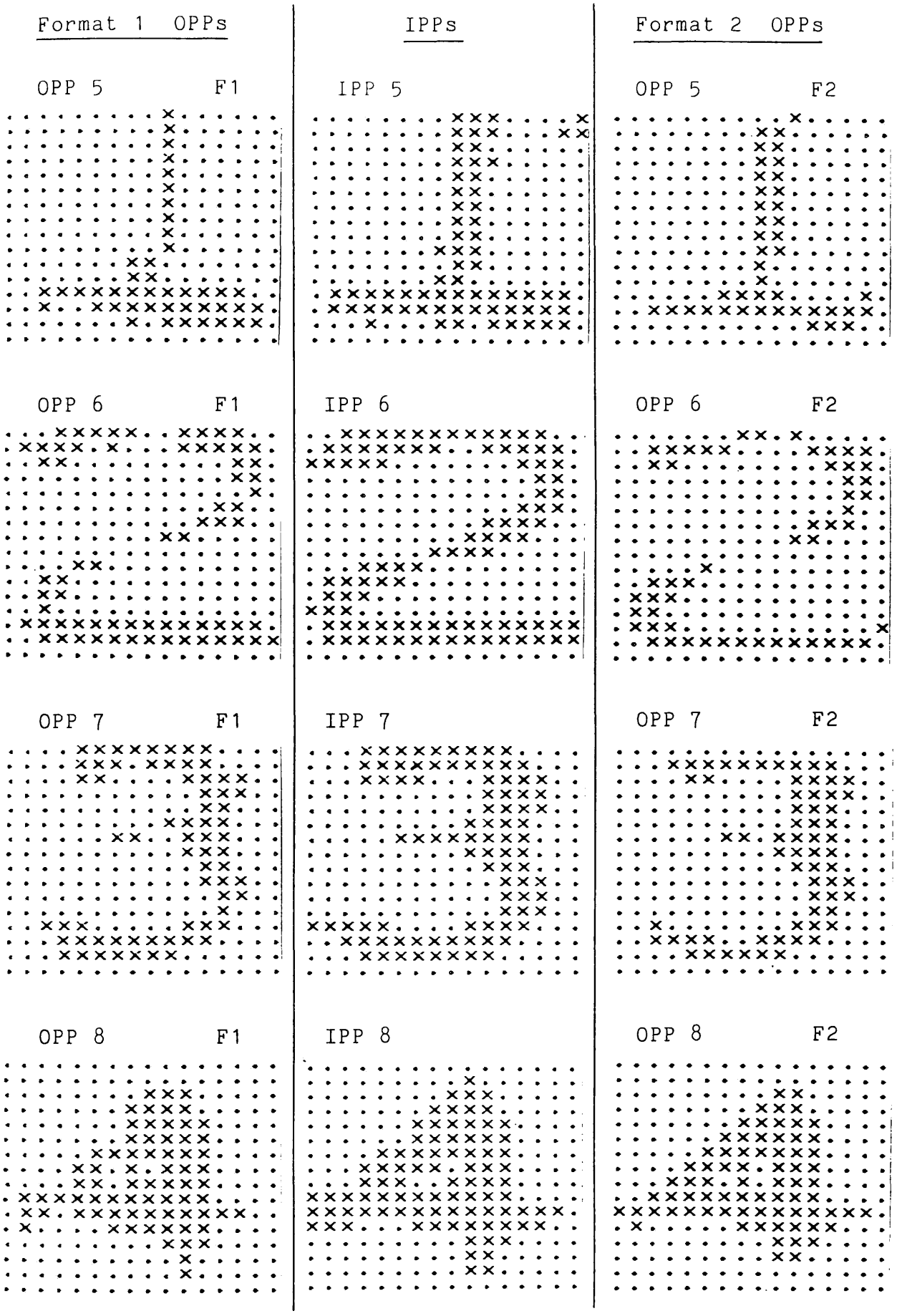


Fig 3.9 Experiment 4 Test Patterns 5-8

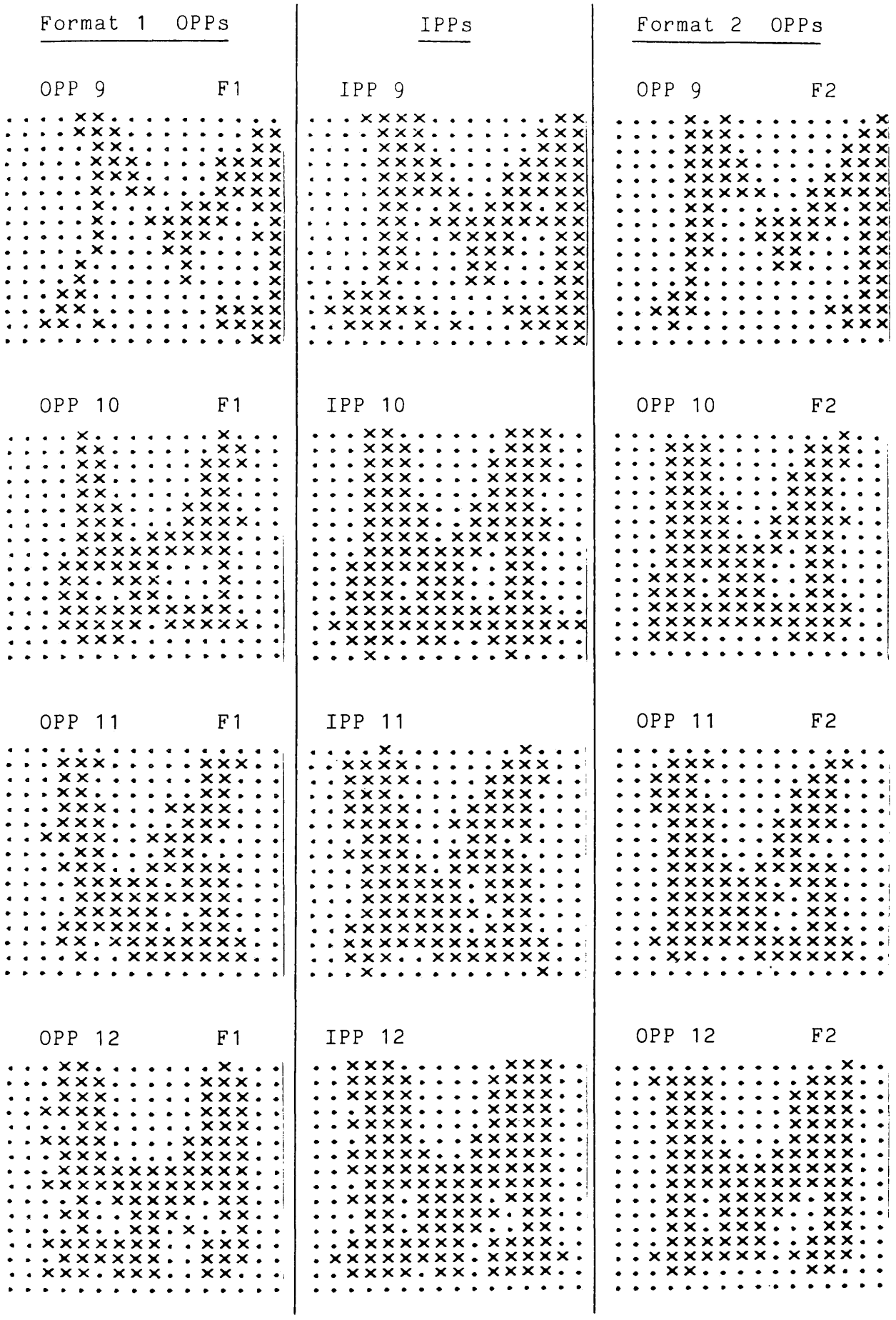


Fig 3.10 Experiment 4 Test Patterns 9-12

significant differences in the performance of the two machines. The test set was composed of three character styles :

1 Four hand-drawn letters 'J' - collected in the same manner as the training set of 'C' and 'D' letters. (IPPs 1-4 in Fig 3.8)

2 Four type-written numerals '1', '2', '3', and '4'. These were collected by the Post Office from actual type-written mail for use in pattern recognition research work (54). (IPPs 5-8 in Fig 3.9)

3 Four type-written letters 'M' from the same Post Office source as the above numerals. These were chosen for their 'unclean' appearance and dissimilarity from the training set to act as a test of the machines' generalisation abilities. (IPPs 9-12 in Fig 3.10)

The output threshold used in the F2 machine in this experiment was set at the mid-range value of '127'. This resulted in a pixel output of '0' if the addressed cell contained '127' or less, or '1' if '128' or more.

Results

A comparison of the left hand (F1) and right hand (F2) columns in Figs 3.8 to 3.10 reveals the variations in performance of the two machines. The most immediately obvious feature of the comparison of the two results is that they appear broadly similar. Both machines do generally thin and clean the characters fed to them in test. This indicates that both F1 and F2 are practicable and perform coherently. (This is a new result with the F2 machine.)

However, on closer inspection it becomes obvious that there are differences which are systematic and repeated between the results of the two formats. These differences are analysed in detail below.

(It should be noted that considerable caution has to be exercised in analysing differences in performance of the two machines, since they are not performing rigorous algorithms; but rather they have been trained on the same rather ad hoc set of characters. Thus, though the results are comparable, it is not possible to place an absolute 'figure of merit' on either performance.)

Discussion

References will be made to specific examples in the test set to illustrate the point being made in the following discussion, although all of the points below generally apply to all of the test patterns.

The F2 machine can be seen to follow the original shape of the object more faithfully and to preserve its outline through the thinning process. This can be seen in IPP 4 and IPP 8 and their respective outputs. In the case of the hooked end of the letter 'J' in IPP 4, this hooked appearance is retained in OPP 4 - F2 but not OPP 4 - F1. The sloping edge of IPP 8 has been stripped back and thinned well, maintaining a straight edge in OPP 8 - F2 unlike that of OPP 8 - F1.

An interesting variation in performance is seen in the handling of vertical lines of two units width. The F1 machine produces single width lines (centred on the right hand of the two original lines); the F2 machine retaining the whole double width line. This can be seen in

IPPs 2, 4, and 5.

The F2 machine also produces smoother vertical edges - resulting in a straighter line, as opposed to the F1 machine's introduction (or retention) of small spurs and nicks in such edges. Examples of these can be seen on the left hand vertical side of IPPs 11 and 12. This does not happen with the right hand edges (IPP 10) where both machines produce a similar result. This discrepancy must be a result of the particular training set used, and hence supplies little useful information regarding the relative performances of the two machines.

Both the machines are capable of breaking a thin character (IPPs 3 and 6) - again as a result of the training set not containing 'correct' examples. As before, the fact that both machines perform essentially identically on this feature conveys little information about their relative merits.

A further point of this type where the machines both act similarly is the removal of noise points, as seen in IPPs 3 and 5.

The F1 machine can be seen to thin features to a considerable degree (often to a unit width limb) whilst retaining the original limb length; whereas the F2 machine appears to strip off the outer layers of such limbs, resulting in thinner but shorter features. This can be seen in the lower limb of IPP 8, and in the bottom of the centre 'V' in IPP 9. In both cases, the F1 machine retained the full length with a single width result, the F2 machine losing some length and not thinning to such a great extent.

These results show that there are some demonstrable improvements in performance when using an F2 machine as opposed to an F1 machine. This difference is primarily due to the fact that the F1 machine simply remembers the last stimulus, giving equal preference to isolated 'bad' examples as to the many 'correct' stimuli. The F2 machine essentially ignores such noise points. However, the improvements in performance are not large, and on the evidence of the present data, it seems safest to conclude that the outputs are 'broadly similar'. The small difference must be viewed in the light of the cost of the more complex machine. This cost can be divided into three categories :

1 Memory Size

The F2 machine uses eight times the amount of memory of the F1 machine. Recall that the F1 machine uses 512x1-bit cells and the F2 machine uses 512x8-bit cells. While it is debatable whether all 256 ($=2^8$) levels are necessary, any machine with even a few levels will require several times the storage capability of the simpler F1 machine.

2 Processing Complexity

The simple F1 machine reads data derived from the EXP directly into the memory matrix cells. These data are later read out - without further processing - to generate OPP. The F2 machine tests the data derived from EXP, and as a result of this test either increments or decrements the contents already contained within the cell. The data, when later read out, has to be compared with a threshold value and the result of this comparison is used to generate the output value used to generate OPP.

3 Processing Speed

As a result of the increased processing required by the F2 machine, the time taken to process a picture will be necessarily longer, both in training and testing.

In the light of these considerations, the overall efficiency of the F2 machine compared with that of the F1 machine is debatable. While some processing improvements have been seen, these have been made at a cost of considerable increase in the processing requirements in terms of size, complexity and time. A compromise solution may, however, be more efficient, wherein more than one threshold is used simultaneously to give more than two output levels. This will be attempted later in Section 5.3 Experiment 7.

3.9 Experiment 5 : Variable Output Threshold and Grey Level Output

The OPPs 1-12 generated by the F2 machine in the previous experiment were all generated using a constant threshold in the testing phase. That is, the contents of each memory matrix cell addressed were compared with a fixed value, the result of this comparison being used to determine the polarity of the output pixel. This threshold was set to the initial value stored in each cell to show a response to as little training as possible.

Direct examination of the internal state of this machine (after training as described previously) reveals that the spread of memory matrix cells extends over all possible values (0-255). Thus, a single threshold value is

not capable of extracting all the information from the memory matrix. This distribution of cells over the possible values before and after training is tabulated in Fig 3.11. For each value, the number of cells containing that value is recorded.

It follows from this spread, that those cells with values far removed from the initial value (ie. values near '0' or '255') have received a larger amount of training than those close to the initial value ('127'). Consequently, the features corresponding to these particular cells have been seen many times in training. As a result, these cells are likely to be reliably set, reflecting accurately the picture processing task being taught to the machine. This measure of 'reliability' of any cell is related to the difference in the cell value before and after training.

The data in Fig 3.11 can be expressed as a histogram to clarify the reliability distribution. (See Fig 3.12.)

From this figure it can be seen that the majority of cells lie close to the original initial value ('127') as they have received few or no stimuli in training. They are therefore likely to be unreliable indicators of the picture processing algorithm taught to the machine. The reverse applies for the two peaks at the extremeties of the range. As a result they are likely to be reliable and the corresponding features are also more likely to occur in testing. This is a pointer to the setting of the threshold value in the test period.

The criteria for setting the threshold value is to obtain the maximum information from the most reliable cells. This is apparently not possible with a single threshold,

Before Training :

Cell Value	No.of Cells
0 ↓ 126	0
127	512
128 ↓ 255	0

After Training :

Cell Value	No.of Cells
0	30
24	1
28	1
30	1
36	2
42	1
50	2
56	1
57	1
60	2
70	2
80	1
87	2
89	1
91	1
94	2
96	2
99	1
101	2
102	1

Cell Value	No.of Cells
103	1
105	1
106	1
107	1
109	2
110	1
114	1
115	2
116	1
117	1
118	1
119	4
120	1
121	3
122	11
123	9
124	18
125	27
126	75
127	198

Cell Value	No.of Cells
128	27
129	11
130	8
131	4
132	3
133	1
134	1
137	2
138	1
139	1
141	2
144	1
146	1
147	3
148	1
151	1
152	1
153	1
154	1
155	1

Cell Value	No.of Cells
156	1
157	1
160	1
163	1
168	1
178	1
188	1
195	1
203	1
227	1
239	1
254	3
255	10

Any other Value in Range 0-255 NOT listed Above :

X	0

Fig 3.11 Distribution of Memory Matrix Cell Values in the F2 m/c before and after Training in Experiment 4

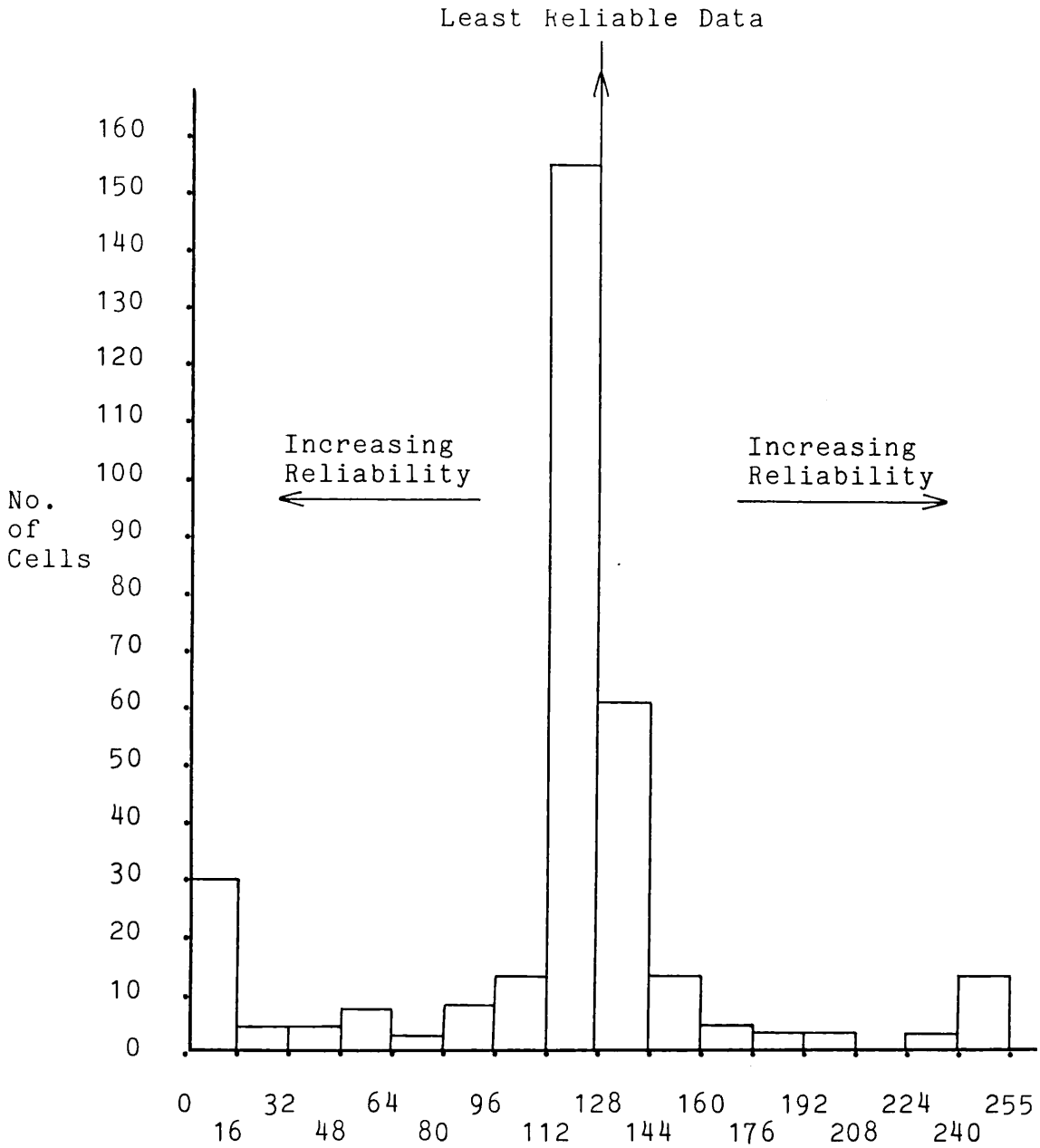


Fig 3.12 Reliability Distribution of Memory Matrix Cells

since there are two maxima of 'reliable' cells either side of a maximum of 'unreliable' cells. Hence a variable threshold was applied in the following experiment to determine the effect on the output. This threshold was swept across the entire range of cell values.

There are two conflicting considerations when attempting to predict the result of sweeping through the

cell values. The first is that the major change in output may be expected when the threshold traverses the central maximum. This is due to the very large proportion of memory matrix cells located in that area, that will change their output as the threshold passes them.

However, on further consideration, a second point emerges, that a large number of these cells are in general unlikely to be accessed at all in testing. As they have not moved far from their original value they were obviously not accessed much in training - the corresponding features were relatively uncommon in the training set. If the test set is similar it will also contain few features that will access these cells remaining near the initial value. As a result, the movement of the threshold causing these cells to output different pixels may not affect the final result greatly. Most of the change in output could be expected at the extremities of the cell value range (around '0' and '255') as these are common cells and are likely to be used more in generating the OPP. Nonetheless, it is important to determine whether there are any configurations that are common, yet appear in the centre of the range, reflecting inconsistent training.

Experiment

An F2 machine was trained on 200 pairs of manually thinned and cleaned 'C's and 'D's, as in Experiment 4. The resultant memory matrix is that which has been examined above. A single test pattern was used - a letter 'J' from the same set of hand drawn characters. The tests were repeated with this pattern several times; each time using a different output threshold. The threshold was swept through

the cell values in steps of 32 (ie. 16,48,80,112,144, 176,208,240). The test IPP 1 together with the resultant set of OPPs '16-240' are shown in Fig 3.13.

Results

Examination of these OPPs reveals that there is a gradual change in output as the threshold scans across the memory matrix values. The largest change in the ratio of cells outputting '0' and '1' pixels occurs as the threshold moves between 112 and 144, the respective outputs being OPPs '112,144' in Fig 3.13. However, these pattern outputs not vastly dissimilar. This would confirm the infrequent occurrence of these cells' features in the test set. The changes here are comparable with the changes in OPPs as the threshold moves to the extremes at the cell values (OPPs '16 to 48'; and OPPs '208 to 240'). In all these cases there are only a few pixels difference between each adjacent pair of pictures.

Fig 3.14 presents the degree of change in the output pictures and the number of memory matrix cells on either side of the threshold as it moves. These quantities are recorded for OPPs '16-240' taken in pairs, the difference in adjacent pairs of pictures being measured in terms of the hamming distance between them.

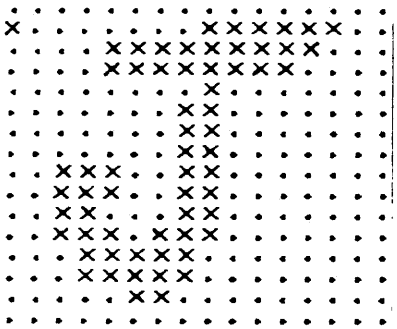
Examination of these data reveals the following facts about an F2 machine : the degree of change in output (as exhibited by actual patterns under test) is in no way related simply to the number of memory matrix cells on either side of the output threshold. The large number of memory matrix cells clustered near the initial value must, for this data, all represent rarely or never found features.

OPPs Produced with a Threshold Varying from

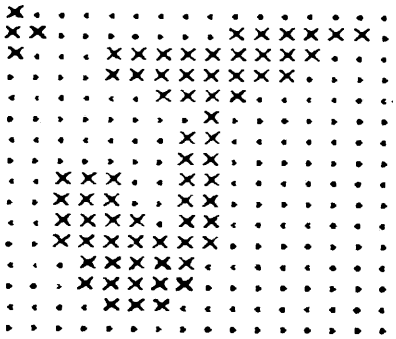
16 - 240

in Steps of 32 units

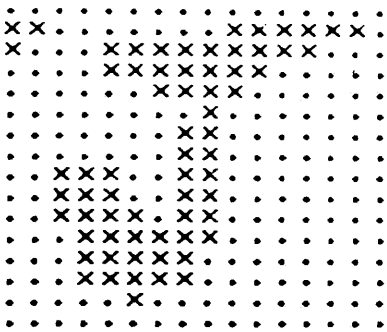
IPP 1 Original



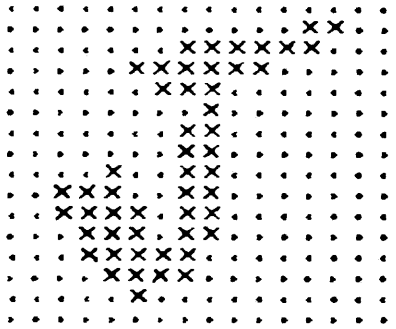
OPP'16' Th=16



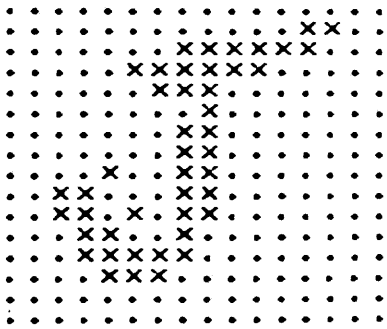
OPP'48' Th=48



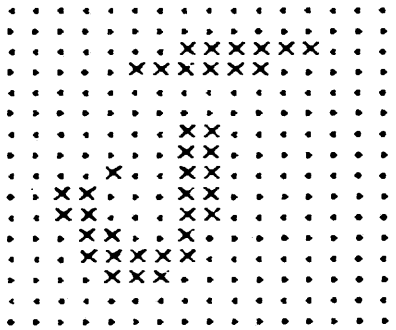
OPP'80' Th=80



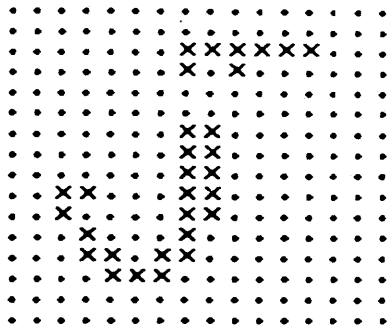
OPP'112' Th=112



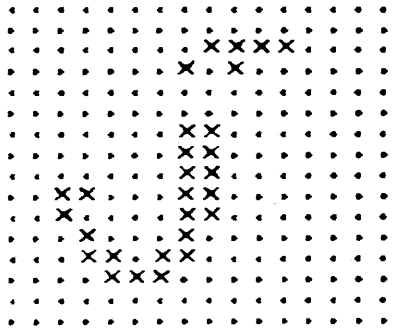
OPP'144' Th=144



OPP'176' Th=176



OPP'208' Th=208



OPP'240' Th=240

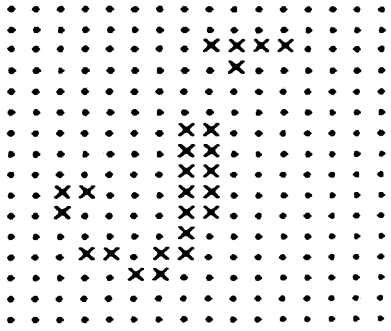


Fig 3.13 Experiment 5 IPP 1 and OPPs '16-240'

OPP Pair (Threshold)	16, 48	48, 80	80, 112	112, 144	144, 176	176, 208	208, 240
No.M.M.Cells Moving Across Threshold	6	8	20	313	16	4	2
Hamming Distance Between OPPs	5	16	6	7	8	2	3

Fig 3.14 Table of OPP Changes against M.M Cells Crossing Threshold

The converse is also true : the small number of cells with values far removed from the initial value are likely to predominate in the outputs, as they represent common features. This indicates that a greater variation could be expected by increasing the amount of training than by moving the threshold.

Continuous (Grey Scale) Output

Examination of the OPPs '16-240' in Fig 3.13 will reveal that this series of binary patterns, generated using an increasing threshold, effectively constitutes a single grey scale pattern. The F2 machine can be used to generate many-valued output pixels in testing, by removing the threshold and comparator of Fig 3.7. In this mode, the binary input pattern addresses cells in the memory matrix which contain digitally continuous values, which are output directly. Examples of such pseudo-grey-scale OPPs are illustrated in Fig 3.15, together with the corresponding (binary) IPPs displayed in the same format. (This format depicts grey levels as '00' for white, 'FF' (hexadecimal) for black, '7F' as the initial value, and intermediate

Binary IPPs

IPP 2

```
00 00 00 00 00 00 00 00 FF FF FF 00 00 00 00 FF
00 00 00 00 00 00 00 00 FF FF FF 00 00 00 FF FF
00 00 00 00 00 00 00 00 FF FF 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF FF 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF FF 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF 00 00 00 00 00 00
00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00
00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00
00 00 00 FF 00 00 00 FF FF 00 FF FF FF FF FF 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

IPP 3

```
00 00 FF FF FF FF FF FF FF FF FF FF FF FF 00 00
00 FF FF FF FF FF FF FF 00 00 FF FF FF FF FF 00
FF FF FF FF FF 00 00 00 00 00 00 00 FF FF FF 00
00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF 00
00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF 00
00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF 00
00 00 00 00 00 00 00 00 00 00 FF FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 FF FF FF FF 00 00
00 00 00 00 00 00 00 FF FF FF FF 00 00 00 00 00
00 00 00 FF FF FF FF 00 00 00 00 00 00 00 00 00
00 FF FF FF FF FF 00 00 00 00 00 00 00 00 00 00
00 FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00
FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00
00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

IPP 4

```
00 00 00 FF FF FF FF FF FF FF FF FF 00 00 00 00
00 00 00 FF FF FF FF FF FF FF FF FF FF 00 00 00
00 00 00 FF FF FF FF 00 00 00 FF FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 FF FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 FF FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 FF FF FF FF 00 00
00 00 00 00 00 FF FF FF FF FF FF FF FF 00 00 00
00 00 00 00 00 00 00 00 00 00 FF FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 FF FF FF 00 00 00
00 00 00 00 00 00 00 00 00 00 00 FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 00 FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 00 FF FF FF 00 00
FF FF FF FF FF 00 00 00 00 FF FF FF FF 00 00 00
00 00 FF FF FF FF FF FF FF FF FF FF 00 00 00 00
00 00 00 FF FF FF FF FF FF FF FF 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Grey-Scale OPPs

OPP 2

```
00 00 00 00 00 00 00 00 32 46 9B 00 00 00 00 7B
00 00 00 00 00 00 00 00 FE FF 69 00 00 2A 72 7F
00 00 00 00 00 00 00 00 FE A3 6D 66 00 00 00 00
00 00 00 00 00 00 00 00 FE 97 79 38 00 00 00 00
00 00 00 00 00 00 00 00 FE FF 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FE FF 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FE FF 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FE FF 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FE FF 00 00 00 00 00 00
00 00 00 00 00 00 2A 5E 82 FF 00 00 00 00 00 00
00 00 00 00 00 00 5B 6A 82 69 00 00 00 00 00 00
00 00 00 00 00 00 00 3C 80 7B 7E 00 00 00 00 00
00 32 46 46 46 46 82 FF FF BC 46 46 46 46 9B 00
00 32 EF 93 FE 9A EF FF FF 99 FF FF FF FF FF 00
00 00 24 57 00 00 24 60 1C 6E 60 9A 9A 9A 79 00
00 00 00 39 00 00 00 00 00 67 00 00 00 00 00 00
```

OPP 3

```
00 00 00 46 46 46 46 86 80 7F 83 46 46 24 00 00
00 3C FF FF FF FE 9A 1C 00 24 60 EF FF FF 8B 00
72 7A 9A 9A 1C 00 00 00 00 00 00 24 C3 FF FF 00
00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF 00
00 00 00 00 00 00 00 00 00 00 00 00 00 FE FF 00
00 00 00 00 00 00 00 00 00 00 00 00 3C FF 69 00
00 00 00 00 00 00 00 00 00 00 00 00 82 FF CB 00
00 00 00 00 00 00 00 00 00 3C FF FE 1C 00 00 00
00 00 00 00 00 00 7E 7F 7F 7A 1C 00 00 00 00 00
00 00 00 00 46 86 7E 7B 00 00 00 00 00 00 00 00
00 32 82 FF FE 1C 00 00 00 00 00 00 00 00 00 00
00 FE FF CB 00 00 00 00 00 00 00 00 00 00 00 00
5E 82 A3 7B 7E 00 00 00 00 00 00 00 00 00 00 00
00 FF FF BC 46 46 46 46 46 46 46 46 46 46 46 9B
00 32 9A 9A 9A 9A 9A 9A 9A 9A 9A 9A 9A 9A 9A 79
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

OPP 4

```
00 00 00 32 46 46 46 46 46 46 46 24 00 00 00 00
00 00 00 FE FF FF FF FE 9A EF FF FF A8 00 00 00
00 00 00 32 9A 9A 1C 00 00 63 FF FF FF 8B 00 00
00 00 00 00 00 00 00 00 00 00 FE FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 FE FF FF 69 00 00
00 00 00 00 00 00 00 00 00 3C FF FF FF 00 00 00
00 00 00 00 2A 7C 80 80 7F 82 FF FF FF 00 00 00
00 00 00 00 00 00 00 00 24 C3 FF FF FF 00 00 00
00 00 00 00 00 00 00 00 00 00 C3 FF FF 00 00 00
00 00 00 00 00 00 00 00 00 00 00 FF FF 8B 00 00
00 00 00 00 00 00 00 00 00 00 00 FE FF FF 00 00
00 00 00 00 00 00 00 00 00 00 5E FE FF 69 00 00
7C 7F 83 46 24 00 00 00 00 00 82 FF CB 00 00 00
00 24 C3 FF FF BC 46 46 82 FF FF CB 00 00 00 00
00 00 00 60 9A 9A 9A 9A 9A 9A 1C 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(Hexadecimal Notation : 256 Levels ;)
(00=White, 7F=Initial Value, FF=Black)

Fig 3.15 Experiment 5 Grey-Scale Output

levels as intermediate numerical values.) The inputs are the same as those used earlier in Experiment 4 (IPPs 5-7, Fig 3.9) as was the training received.

If the two effective grey-levels used in the case of the binary patterns are assumed to lie at the extremities of the grey-scale (pure black and pure white) then this processing constitutes a form of low-pass spatial filtering. As such, it may be of use in certain picture processing and scene analysis applications (40). It is effectively the inverse transformation of 'thresholding' in the normal sense when applied directly to a grey-scale digitized pattern.

This production of thinned and cleaned grey-scale pictures, from a machine having been trained only on binary inputs forms an interesting investigation. Similar techniques using 'fuzzy logic' have been employed elsewhere to handle grey scale pictures with binary processes (30).

3.10 Summary of Experiments 1 - 5

The preliminary experiments described in this chapter have shown an LPP machine to be a workable proposition. While this has been distinctly encouraging, and a spur to further investigation, the results have not always been clear cut, and the rather simple experiments tried have not yet revealed final answers as to how to set up LPP machines. For this reason Chapter 4 analyses the underlying theory and possibilities in some detail, and Chapters 5 and 6 describe further, more rigorous experiments on LPP machines. These later chapters also move towards real applications by employing pictures with rather more than 16^2 resolution. In

addition, sequential as well as parallel processing will be attempted.

CHAPTER 4

THE GENERAL LEARNING PICTURE PROCESSOR

4.1 An Introduction to the Development of a General LPP Machine

The experimental work performed so far on the learning picture processor has suggested some departures from the original machine as conceived in Chapter 2. These changes have been in response to the need for improvements in performance, but have been arbitrary. It is possible to create a more formalized and structured concept of a general learning picture processor. This framework will also aid the generation of new variants, as a large family of machines can be proposed. Evidently, the resulting large numbers of machines cannot all be analysed by experiment within the confines of one thesis. However, predictions of performance can be made, based on theoretical considerations and extrapolations from the modest machines described in earlier chapters. There are several levels of development in this general machine. These include :

1 Variations in the internal data processing of a single, simple LPP machine

2 Different modes of operation of such variants in terms of the handling of test picture data

3 The use of feedback around a single stage machine

4 The cascading of machines

These may in principle all be combined to produce large and complicated machines. Such machines would be exceptionally complex to analyse, and as such may well illustrate the limit of the trainable picture processor type of machine. The following sections discuss these levels of development in more detail.

4.2 Variations in the Internal Data Processing of the LPP Machine

This section proposes a generalized system by which a LPP machine may operate. Implied by such a generalized system are variations in the means by which :

1 useful information is extracted from the training patterns;

2 this data is used to modify a memory matrix to reflect the task to be learnt;

3 data in the memory matrix is used to generate subsequent output pictures in testing.

Examples of Data Transformation through 'Function Modules'

At each internal stage of such a machine, there is a transformation of data. Some examples would be :

1 The extraction of a window of pixels according to the input picture and the current values of the x and y co-ordinates,

2 The transformation of this window into an address of a cell in the memory matrix.

From these, it can be seen that the data processing occurs as a series of transformations which can be represented as a series of interconnected 'function modules'. Each module takes data inputs and performs a particular function upon them, thus generating output data. In the case of the above two modules, their functions could be represented as :

1 $W = f_a(IPP , x , y)$

2 $A = f_b(W)$

where : IPP represents the binary input picture,

x , y the cartesian co-ordinates of the window,

W the window contents,

A the memory matrix address,

and f_a and f_b the functions to be performed on the arguments above.

It is a change in these functions (f_a and f_b above) that represents a design change in the machine in general terms. Fig 4.1a shows how such function modules will be represented both algebraically (as above) and diagrammatically below.

Function Module Representation of the General LPP Machine

The whole internal operation of the machine can be represented as a set of these function modules. Such a machine is illustrated in Fig 4.2, nomenclature being as defined in Fig 4.1b.

(a) Function Modules :

Function expressed algebraically as

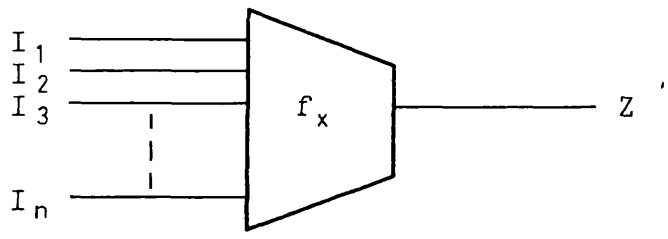
$$Z = f_x(I_1, I_2, I_3 . . . I_n)$$

where : f_x is the function

I_1 to I_n are the independent input variables

and Z is the dependent output variable

Functions will be expressed diagramatically in Fig 4.2 as :



(b) Nomenclature of Variable Names to be used in Fig 4.2 :

IPP	Complete Input Picture (to be processed)
EXP	Complete Example Picture (used in Training)
OPP	Complete Output Picture (generated in Testing)
x	Horizontal) Cartesian Co-ordinates of
y	Vertical) Current Scan Position
W	Window Contents Extracted from IPP
A	Address of Memory Matrix Cell
MM	Original Complete Memory Matrix
mm(A)	Original Single Memory Matrix Cell addressed by A
mm'(A)	Updated Single Memory Matrix Cell addressed by A
MM'	Updated Complete Memory Matrix
P0	Centre point pixel in Window defined by x , y
P1-P8	Immediate eight-connected neighbours of P0

Fig 4.1 General Function Module Description and Variable Nomenclature

Algebraic Representation :

1	$W = f_1(\ IPP\ ,\ x\ ,\ y\ )$	Input Window Extractor
2	$A = f_2(\ W\ )$	Address Calculator
3	$mm(A) = f_3(\ MM\ ,\ A\ )$	Cell Extractor
4	$EXP/PO = f_4(\ EXP\ ,\ x\ ,\ y\ )$	Example Window Extractor
5	$mm'(A) = f_5(\ EXP/PO\ ,\ mm(A)\ )$	Cell Modifier
6	$MM' = f_6(\ mm'(A)\ ,\ MM\ )$	Cell Replacer
7	$OPP/PO = f_7(\ mm(A)\ )$	Output Pixel Generator
8	$OPP = f_8(\ OPP/PO\ ,\ x\ ,\ y\ )$	Output Picture Generator

Diagrammatic Representation :

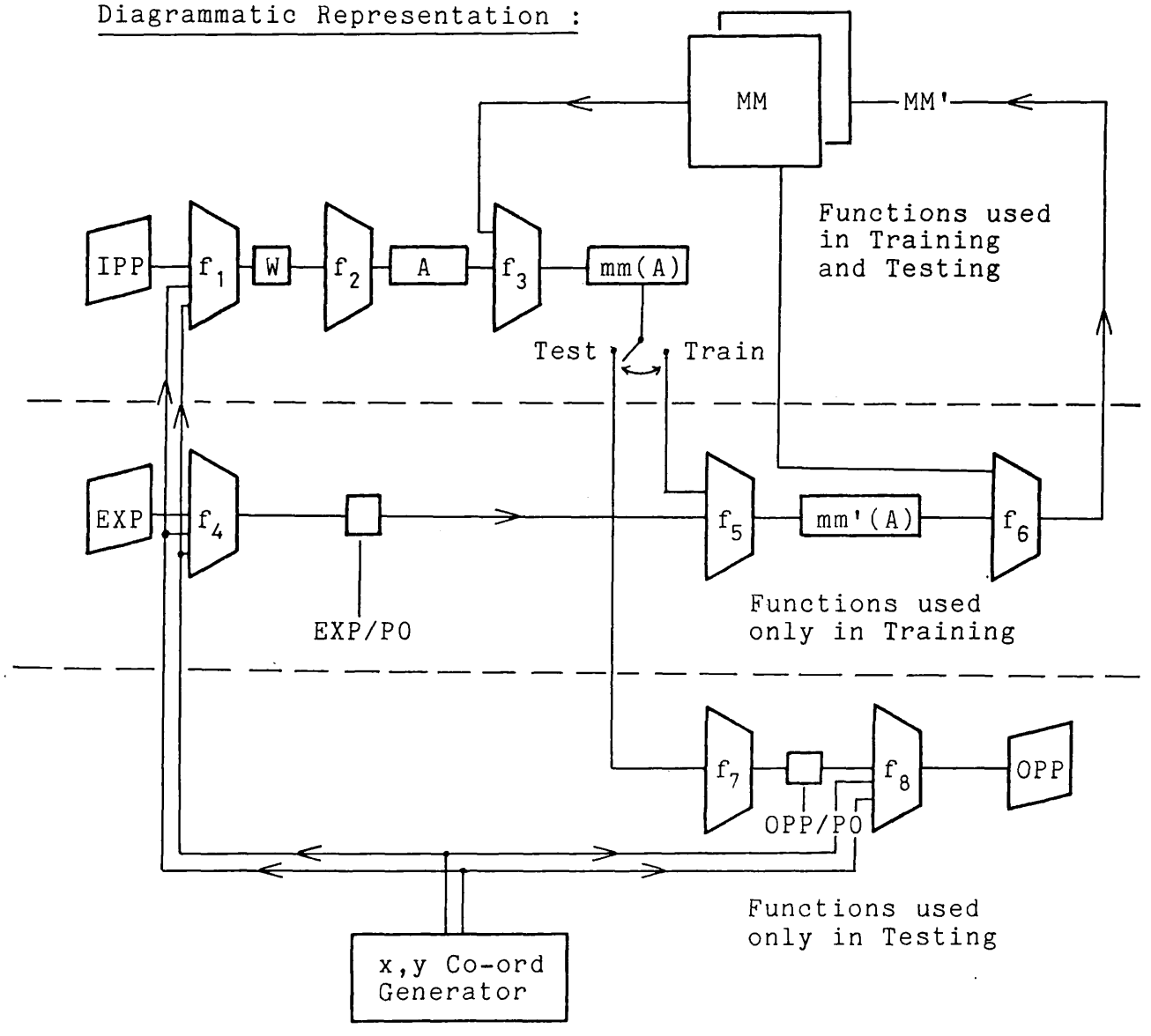


Fig 4.2 Internal Function Modules in a General LPP Machine

When the functions themselves are specified, the machine is transformed from a general one to a particular variant. To illustrate this, the definition of the F1 machine of Chapter 2 will be considered.

Specific Definitions of General Functions for F1 Machine

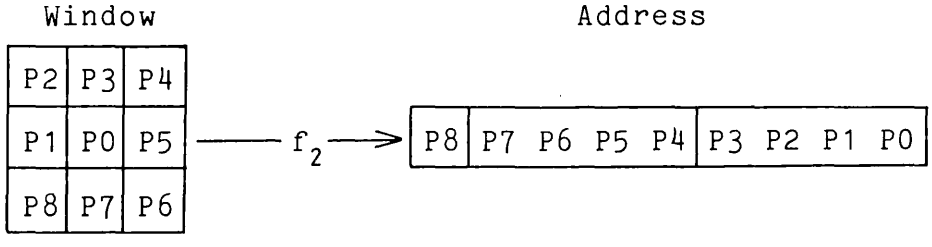
For each of the eight functions in Fig 4.2, there follows a general explanation and a specific definition related to the F1 machine described earlier :

1 Input Window Extractor $W = f_1(IPP , x , y)$

This is effectively the scanning window extractor (WE I) of Fig 2.8 acting on the input picture. This device takes a pair of co-ordinates x,y in the range 0-15 to define a centre point and its adjacent eight-connected neigbours in the input picture. These nine pixels (P0-P8) form the window contents W.

2 Address Calculator $A = f_2(W)$

This function re-arranges the window contents W to form the 9-bit binary address - in this case the exact format of the re-arrangement is arbitrary, but follows the pattern below :



(Both 9-bit binary variables)

This function (in this example) merely re-formats the data.

3 Cell Extractor $mm(A) = f_3(MM , A)$

This function extracts the single memory matrix cell pointed to by the address generated above. This is simply the accessing of a particular bit of storage in a table of 512 bits.

4 Example Window Extractor $EXP/PO = f_4(EXP , x , y)$

This is the scanning window extractor (WE II) of Fig 2.8. This extracts the centre point PO from the example picture according to the current value of the x and y co-ordinate generator. This pixel becomes EXP/PO. (This function only operates in training.)

5 Cell Modifier $mm'(A) = f_5(EXP/PO , mm(A))$

This is the modification of the addressed cell according to the value of the centre point extracted from EXP/PO. This produces the new cell value, which, in this simple example is set equal to EXP/PO. (This function is also only active during training.)

6 Cell Replacer $MM' = f_6(mm'(A) , MM)$

This is the updating of the memory matrix as a whole, after each training stimulus has been received into the particular cell. Here, this is simply the replacing of the cell (with its modified contents) into the table in its correct position - the writing back into the RAM. (Training only.)

7 Output Pixel Generator $OPP/PO = f_7(mm(A))$

This is the generating of a new output pixel from the cell contents, for insertion into the OPP field. In the simple example taken this OPP/PO is set equal to the value

of the addressed cell. (This operates only in the testing phase.)

8 Output Picture Generator OPP = f_8(OPP/PO , x , y)

This represents the picture generator (PG III) of Fig 2.8 and is the generation of OPP, pixel by pixel, as each value is computed by the LPP machine. (Testing only.)

This completes a description of the internal functions of the machine when applied to a particular variant - the F1 machine. To facilitate the description of how other variants are defined by changes in these eight functions, they are tabulated for two machines (F1 and F2) in Fig 4.3 below. (It will be noticed that they differ only in functions f_5 and f_7).

If Fig 4.3 is examined in conjunction with Fig 4.2 it will be seen that these functions define the two machines whose internal formats were illustrated in Fig 3.7.

Further Possibilities for Variations

A set of more complex machines may be devised by a systematic examination of the above functions and speculation of the range of possibilities available at each stage. Even the arrangement between these functions shown in Fig 4.2 can be altered to give yet more variations.

For example, a system could be envisaged where the scanning window does not move over the input picture in a regular manner. The window could follow some feature of the pattern itself : eg. an edge that has been defined as the 'driving' data, directing the generation of the x and y co-ordinates of the window. The values x and y would then

General Function	Particular Functions	
	F1 machine	F2 machine
$W=f_1(IPP,x,y)$ Input Window Extractor	W becomes equal to the 9 pixel values at points in IPP defined by $x\pm1, y\pm1$, ie. centre point and its eight immediate neigbours	
$A=f_2(W)$ Address Calculator	A formed by stacking two dimensional binary window into a one dimensional binary address	
$mm(A)=f_3(MM,A)$ Cell Extractor	mm(A) becomes equal to cell in MM with address A	
$EXP/PO=f_4(EXP,x,y)$ Example Window Extractor	EXP/PO becomes equal to pixel xalue at point in EXP defined by x,y	
$mm'(A)=$ $f_5(EXP/PO,mm(A))$ Cell Modifier	mm'(A)=EXP/PO	if EXP/PO=1, mm'(A)=mm(A)+1; if EXP/PO=0, mm'(A)=mm(A)-1
$MM'=f_6(mm'(A),MM)$ Cell Replacer	new cell contents written into MM on top of old contents, remainder of MM unaltered	
$OPP/PO=f_7(mm(A))$ Output Pixel Generator	OPP/PO=mm(A)	if mm(A)>Threshold, OPP/PO=1; if mm(A)$\leqslant$Threshold, OPP/PO=0
$OPP=f_8(OPP/PO,x,y)$ Output Picture Generator	OPP/PO inserted into OPP field at position defined by x,y, remainder of pattern unaltered.	

Fig 4.3 Functions (f_1- f_8) defined for F1 and F2 Machines

themselves be a function of IPP, rather than cycling through all possible values autonomously as at present.

This could be formalized as :

$x = f_9(\ IPP\ )$ x Co-ordinate Generator

$y = f_{10}(\ IPP\ )$ y Co-ordinate Generator

This would obviously affect all the other functions defined so far, as x and y appear in these functions either implicitly as independent variables, or explicitly by substitution.

For a coherent operation of the scanning window extractor, the above functions f_9 and f_{10} could be formulated in a machine programming style as suggested below. The function f_9 might adjust the value of x such that the window always produced a 'crossing number' of two. A similar, and interacting function f_{10} might generate y such that the window scanned along the edges of objects in the field of view.

This is a somewhat simplified suggestion of a rather more complex development of the internal workings of the LPP machine. It serves to illustrate that although the above formulation of LPP variants is comprehensive, it is not exhaustive. It provides one example of a framework upon which a system of machine variants can be built.

Down-loading the Memory Matrix

An important variation that can be expressed in terms of these functions is the down-loading of the memory matrix contents directly from a host machine. The LPP machine is trained by direct injection of data into all of the memory matrix cells, rather than by the generation of these cells

through training by example.

The functions above used in training (f_4, f_5 and f_6) are no longer implemented, and the variable MM - the contents of the complete memory matrix - is introduced from an external source into the system as a set of constants. For the value of MM to be meaningful, it is necessary to be aware of how the functions used in testing (f_7 and f_8) will operate. An example of this use of a down-loaded memory matrix is shown later in Experiment 12.

4.3 The Handling of Picture Data in the Testing Phase

All the machines described so far have been parallel (as described earlier in Section 1.3.3). The possibility of testing 'sequentially' also exists: the two methods of generating the OPP are illustrated in Fig 4.4.

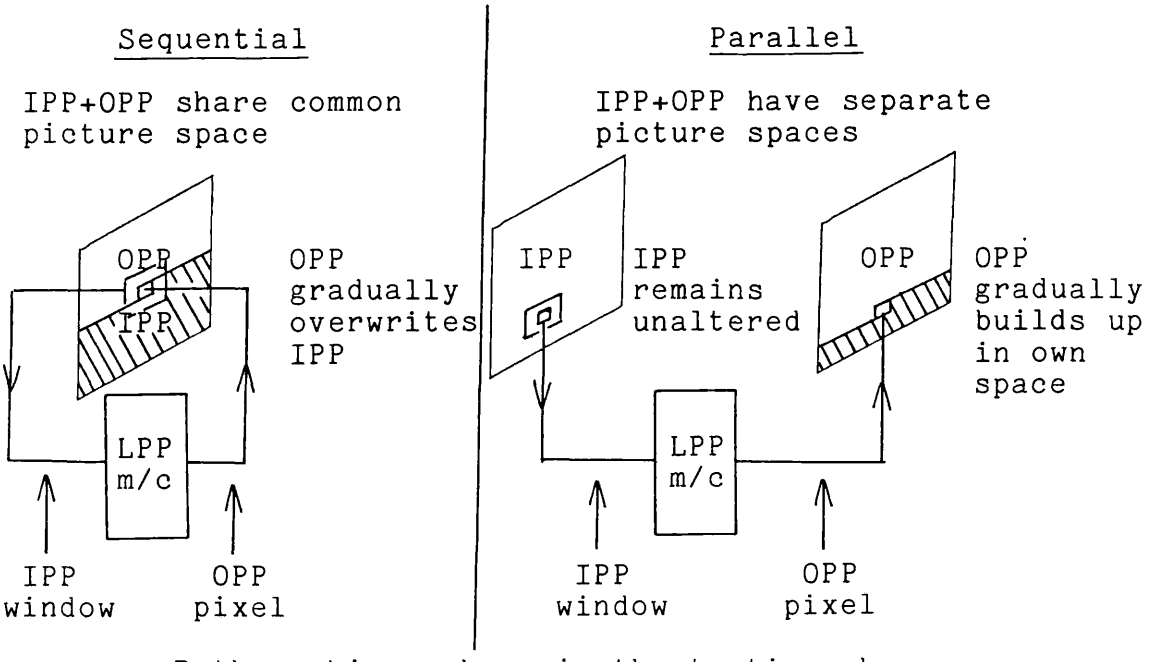


Fig 4.4 The Operation of a Picture Processor in Sequential and Parallel Modes

The Effect of the Different Modes

The difference in effect of these two modes of operation can vary depending on the actual picture processing operation to be carried out. For example, if a simple operation such as inversion is attempted, both modes are likely to generate similar results. However, if a process such as thinning (which is intended to preserve connectedness) is attempted with the same training algorithm, the results are likely to be significantly different. The difference occurs because pixels are processed individually, and would not be removed if doing so would break the connnectedness of a limb. In the parallel mode, as all pixels are processed simultaneously and independently, connectedness might well be broken by removal of pixels from a limb of two units width. As a 'limb-pixel' is processed, it is assumed to still have a neighbouring line of pixels and hence it is correctly removed, but this being true for those neigbouring pixels as well, the entire limb may be removed. Hence the result may be quite different for sequential and parallel modes of operation. The two modes will be compared later on the LPP machine in Experiment 13.

4.4 Feedback around a Single Machine

The concept of feedback of pictures round these machines may be usefully employed. The 'single pass mode' in both training and testing has been implied so far, as illustrated earlier in Fig 2.10. Here, each picture passes through the machine only once. The use of feedback, where

pictures are repeatedly passed through the machine, may be employed in the testing phase.

Assuming a LPP machine is already trained then the test operation, which results in the generation of one output picture from one input picture, lends itself well to this mode of operation. The diameter-limited restriction (see the discussion in Section 3.5) can be lifted also, or at least partially relaxed, dependent on the number of feedback passes to be made. That is, the machine can react to and process features that are spread over a distance greater than that seen by a single window at once. For example, if the machine is attempting to find the centres of objects larger than the window size, this could not be effected by a single pass through the machine. However, multiple passes could progressively strip the outer layers from the objects until the machine is finally left with the centre points as required.

The operation of the feedback mode in testing is illustrated in Fig 4.5. The number of passes made through the machine may clearly vary from one upwards without limit.

There are several criteria by which it could be deemed that sufficient passes have been made. A fixed number of passes could be made (known by previous experience to be sufficient) or the pattern could be passed continually through the machine until no further changes occur in the resultant output. Both these approaches have been employed experimentally (see Chapter 6).

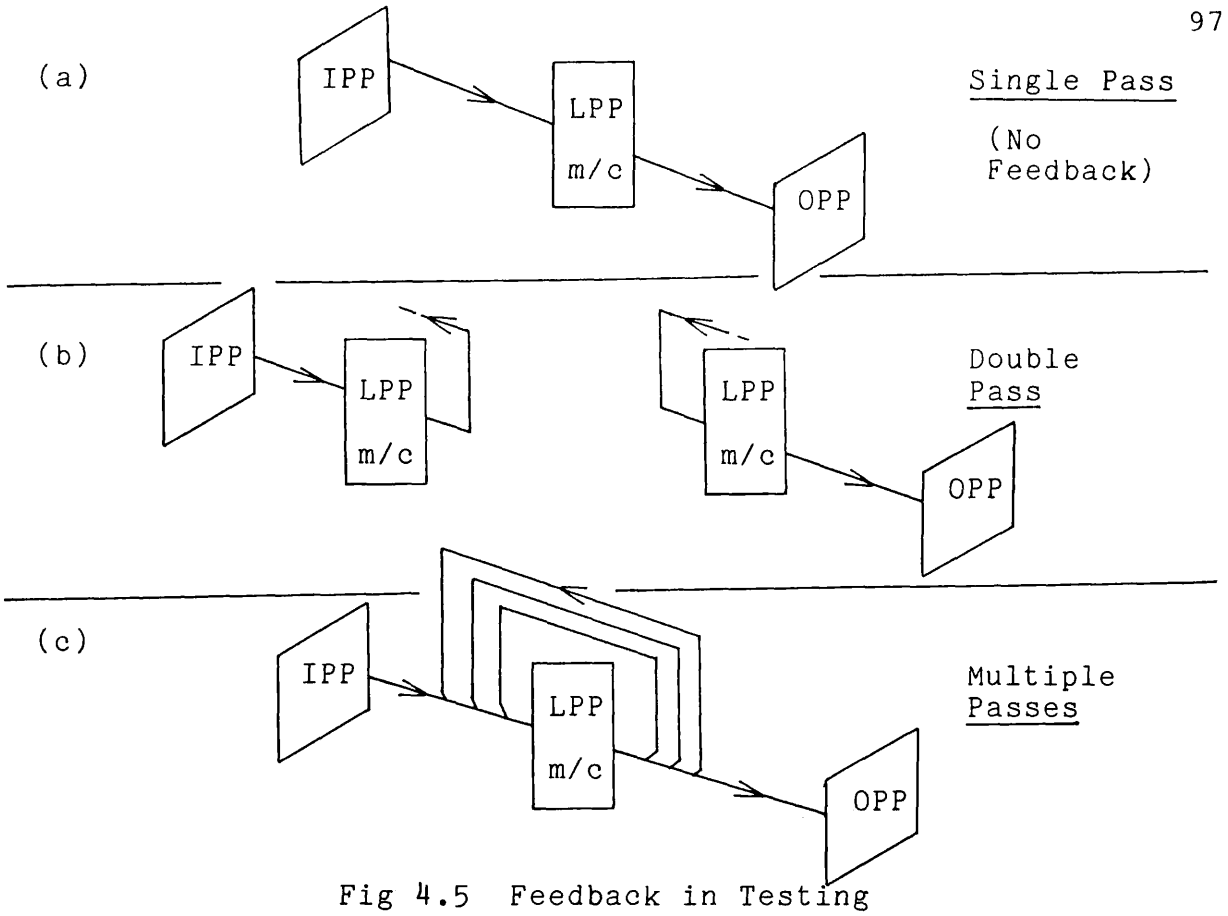


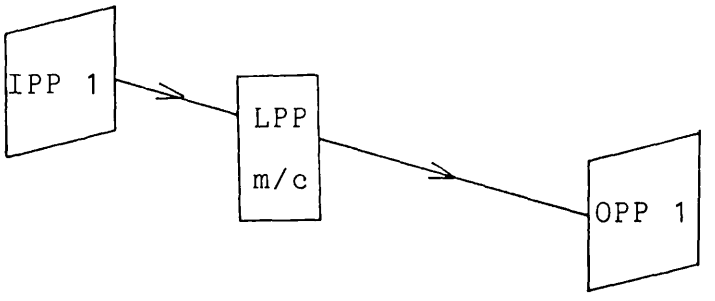
Fig 4.5 Feedback in Testing

4.5 Cascaded Machines

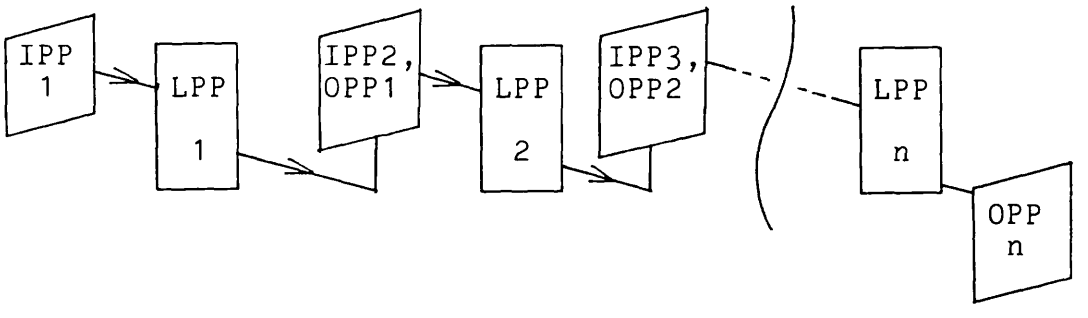
As with any machine that produces output data in the same format as the input data, a set of LPP machines can be cascaded. That is, the output of one machine can be fed into the input of the next, the complete set of machines now being regarded as the whole picture processor. This is illustrated in Fig 4.6b.

Again, as with the case of feedback, this mode of operation may only usefully be employed in the testing phase, where the machine stages each produce an output. In training, as each stage requires two inputs and produces no output there is no obvious method of cascading the machines. Such cascaded stages must therefore be trained independently from each other, and can only be cascaded to commence

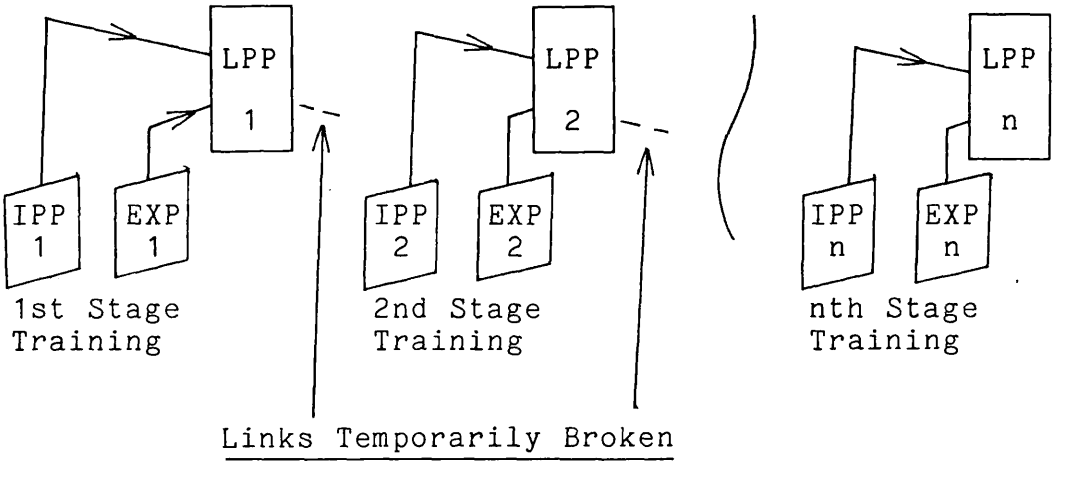
(a) Testing - Single Stage Machine



(b) Testing - Cascaded Machines



(c) Training - Cascaded Machines



Each Stage (1,2 and n shown) represents a particular section of the Transformation from IPP 1 to OPP n

Fig 4.6 Testing and Training of Cascaded Machines

testing. Consequently, the training of each stage must represent the particular (and probably different) process required at each stage of the transformation. This requirement is illustrated in Fig 4.6c.

4.6 Compounded Variants : The Problems of Complex LPP Machines

It has been shown in the above sections how variations can be introduced and to some extent formalized in the LPP machines. These variations have been broadly divided into four major classes :

1 internal data processing,
2 parallel or sequential mode of operation on pictures,
3 the use of feedback around a machine, and
4 the cascading of machines.

These last two classes together introduce the idea of compounded machines, where feedback need not only occur around single stages but possibly around more than one stage.

In principle, feedback of data can occur any number of times around any number of stages, and this represents the compounded machine both cascaded and with feedback. If the individual stages are capable of assuming any of the internal or mode variations described earlier, then this represents a more general machine.

However, this concept of the large and general machine is of questionable practical use. A machine that exploited all of these options simultaneously might be impossibly

complex to analyse by experiment. Thus it appears reasonable to let such complexities evolve as necessity dictates.

In the experiments that follow, the machines used are thus generally simple variants, to enable meaningful comparisons of performances. At this early stage in the development of trainable picture processing systems, these experiments are aimed at producing results that can be extrapolated to more complex systems later.

CHAPTER 5

EXPERIMENTS WITH MORE ADVANCED LPP MACHINES

5.1 An Introduction to Further Experimental Work

As a result of the preliminary experiments in Chapter 3 some practical modifications have been suggested for the basic LPP machine. A general set of theoretical modifications has been described in the previous chapter, and some more experimental variations will be attempted here.

It will be noted that this is not primarily an investigation of the picture processing tasks themselves, rather an investigation into how the LPP machine performs such tasks, and the effect of design and operational changes on this performance. Hence a fixed range of picture processing tasks will often be retained over several of the experiments to permit comparative judgements of the results.

In performing these experiments it will be appreciated that there is no easy way of accurately or rigorously measuring the performance. Consequently, there will be a heavy reliance on visual examination of the picture outputs for assessment of the results.

5.2 Experiment 6 : A Range of Picture Processing Tasks

In the experiments performed so far, only one picture processing task has been attempted. This was the manual 'tidying up' of alphabetic characters involving cleaning and

thinning. To exercise the system on other tasks, an LPP machine was trained to perform four different tasks. These were : cleaning, inverting, thinning and thickening of characters.

The machines used so far have operated on 16^2 patterns as these were adequate to show up the points raised regarding the LPP system. However, in the following experiments the picture resolution will be doubled - 32 samples will be taken in each direction. The 16^2 machine was defined as format 'F1'; the 32^2 machine is referred to as the 'F3' machine. It is otherwise identical internally, with two-levelled memory matrix cells - initially set to '0'.

Train/Test Cycle

One IPP+EXP pair of patterns was found here sufficient to train the machine, and one test IPP pattern sufficient to establish the performance in general terms. This is because each 32^2 pattern effectively generates 1024 training patterns when used with a scanning window device and consequently trains the machine to a considerable extent.

For each of the four tasks to be learnt : clean, invert, thin, thicken; this IPP+EXP pair is shown at the top of a block of 4 patterns in Figs 5.1 to 5.4. Below these two patterns are shown the test IPP and the resultant OPP generated by the machine working in the parallel mode.

Results

The OPPs in Figs 5.1 to 5.4 illustrate that this machine can learn different tasks. While there is an unavoidable requirement of greater frame storage, the actual Learning Picture Processor is internally the same size for

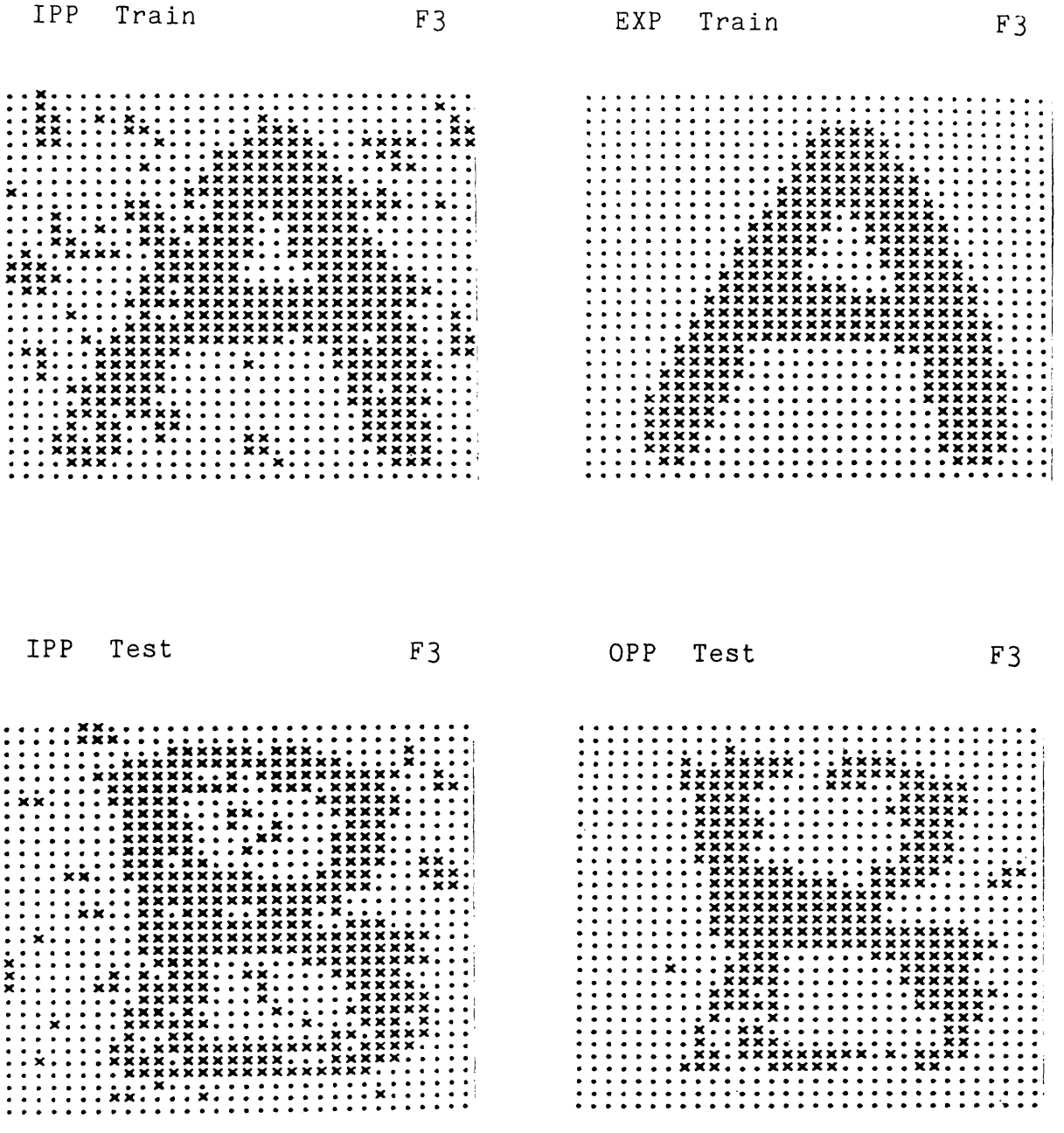
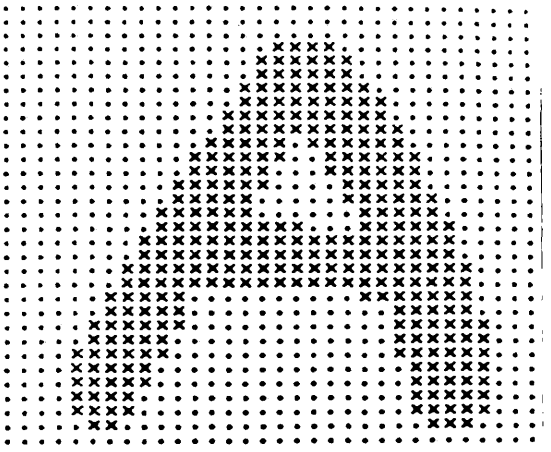
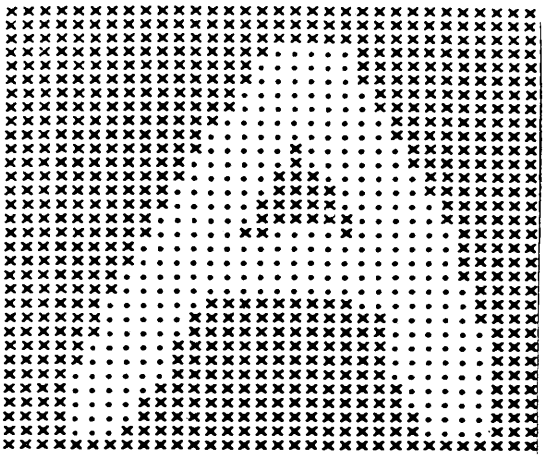


Fig 5.1 Experiment 6 Task : CLEAN

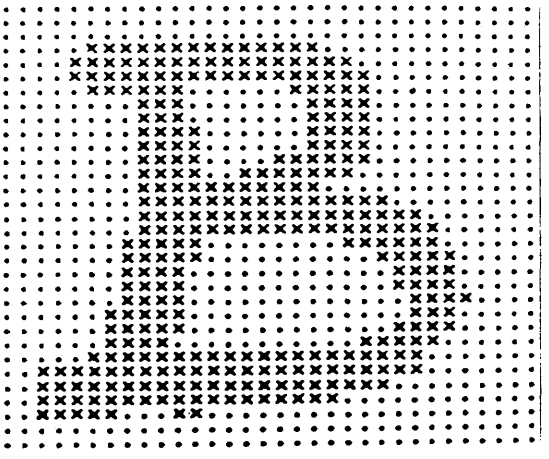
IPP Train F3



EXP Train F3



IPP Test F3



OPP Test F3

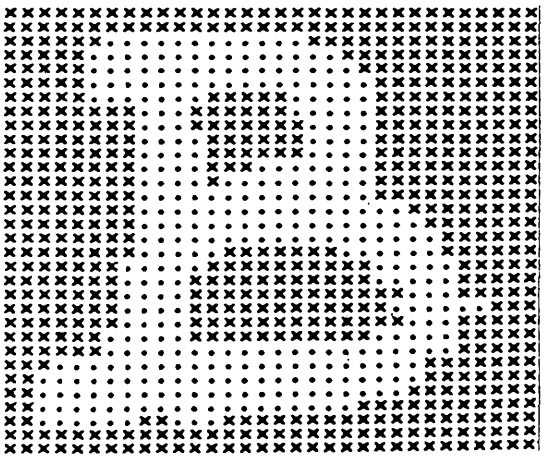
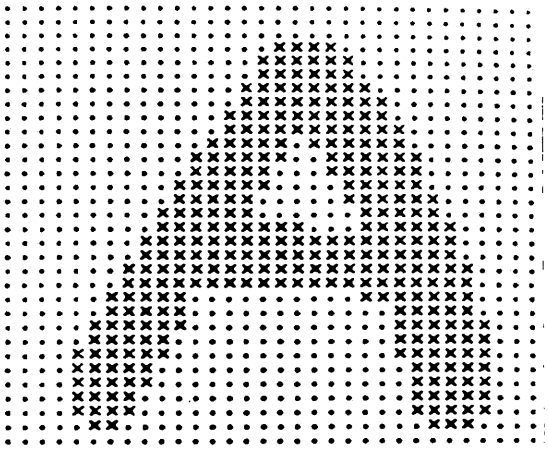
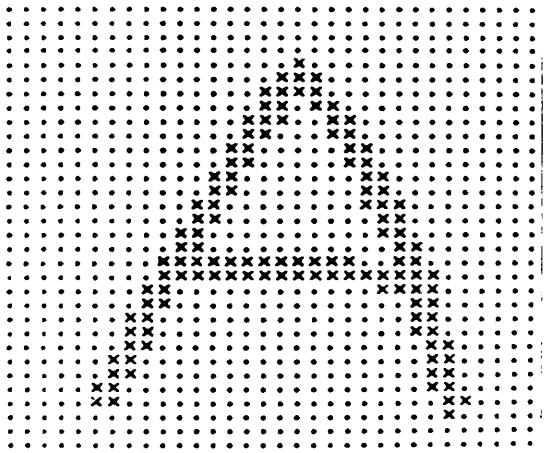


Fig 5.2 Experiment 6 Task : INVERT

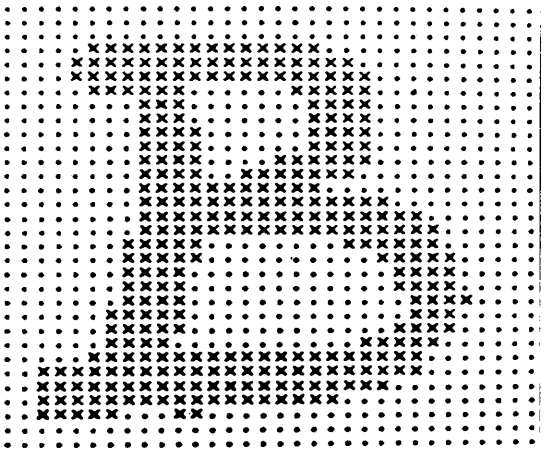
IPP Train F3



EXP Train F3



IPP Test F3



OPP Test F3

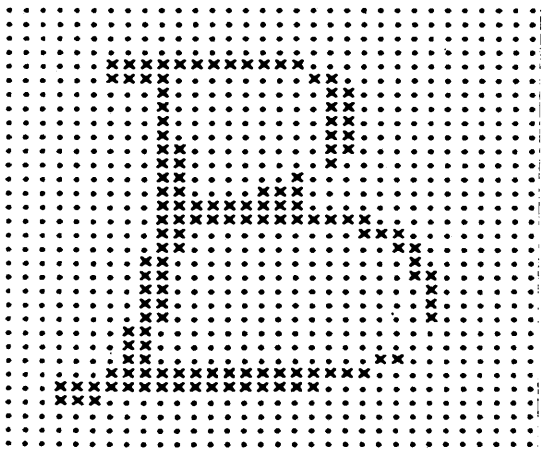
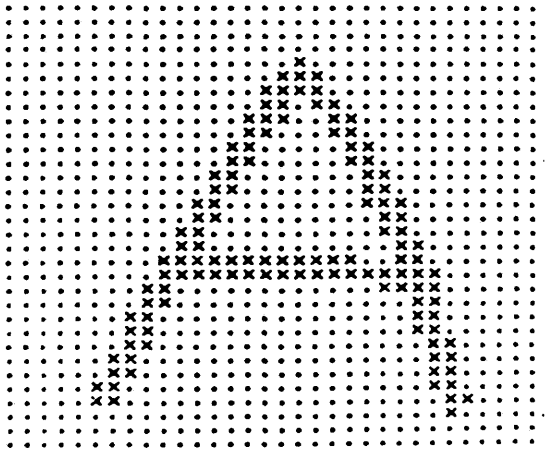
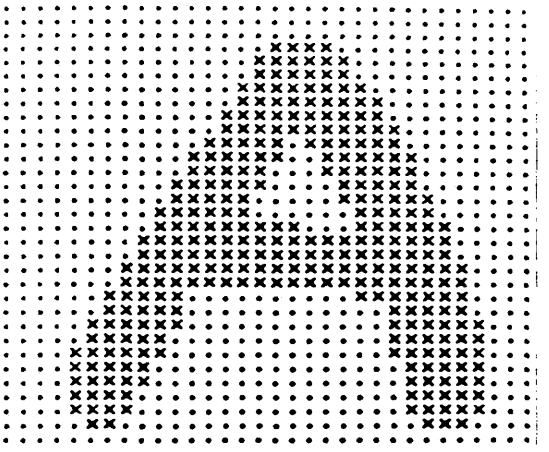


Fig 5.3 Experiment 6 Task : THIN

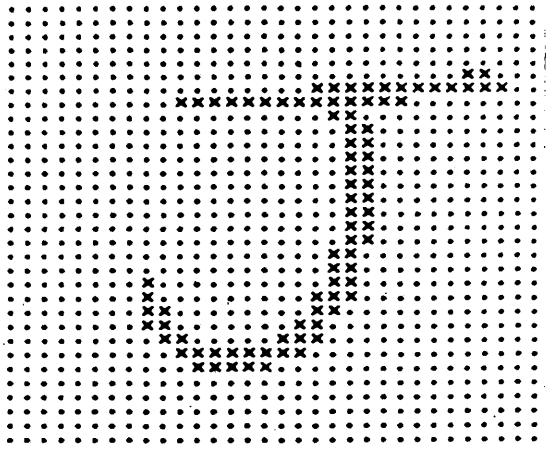
IPP Train F3



EXP Train F3



IPP Test F3



OPP Test F3

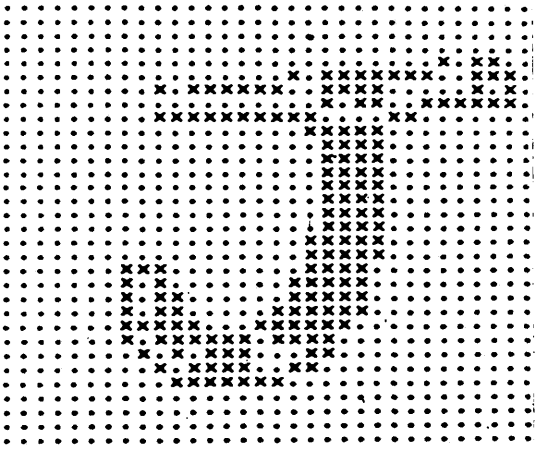


Fig 5.4 Experiment 6 Task : THICKEN

both the 16^2 and 32^2 formats. This is an important point - since both machines use a scanning window, which effectively serializes the input picture; a machine of fixed size could process a picture of any size given sufficient time.

In the experiments that follow, the 32^2 picture is taken as the current standard for the LPP machine, which has here been shown capable of learning different picture processing tasks as a result of re-training alone.

5.3 Experiment 7 : Tri-state/Bi-state Memory Matrix Cells

It has been noted in Experiment 5 that the use of multi-valued memory matrix cells (the F2 machine) may lead to possible improvements in performance, by extracting more of the useful information from the memory matrix. However, with the F2 machine suggested earlier, the complex internal processing is an expensive price to pay for the apparently small gains in performance.

A compromise should be possible with the use of a smaller range of memory matrix cell values. Ideally, this could also simplify the processing requirement and hence further improve the machine's efficiency.

It is suggested here that just three cell values would retain in the memory matrix most of the useful information available in the training set. Referring to Fig 3.11 it can be seen that the largest single group of memory matrix cells remains at the initialisation value - even after the machine has received considerable training. The only useful information contained within these cells is that the features they represent have not been seen in training. If a

test pattern selects one of these untrained cells, by presenting a feature not seen before, it would be advantageous if the resultant output pixel could indicate this lack of training. The resultant F4 machine would operate in the following manner.

Before training, all cells are preset to the initialisation value, represented as '?'. During training, the stimuli received from the training set cause the cells addressed to be either set to '1' or reset to '0'. In testing, those cells that have not been trained at all (and hence still contain '?') can now be distinguished from those that have been trained and contain either '0' or '1'. An appropriate output can then be made into each OPP pixel to reflect this. Repeated training in the same cell, but in the opposite sense will immediately overwrite the old cell contents with the new value, without re-entering the '?' state.

The operation of the F4 machine is represented in Fig 5.5 in terms of the response of the memory matrix to training stimuli. The F4 machine's response is shown compared to that of the F3 machine as used earlier. (The F3 response is shown for the two possible initialisation values.)

It is clear from Fig 5.5 that the F4 (Tri-state) memory matrix will retain all the information regarding the training stimuli that is found in both cases of the F3 machine's initialisation.

Experiment

This situation is illustrated by the test run shown in Fig 5.6 comparing these machines. This test was run in the

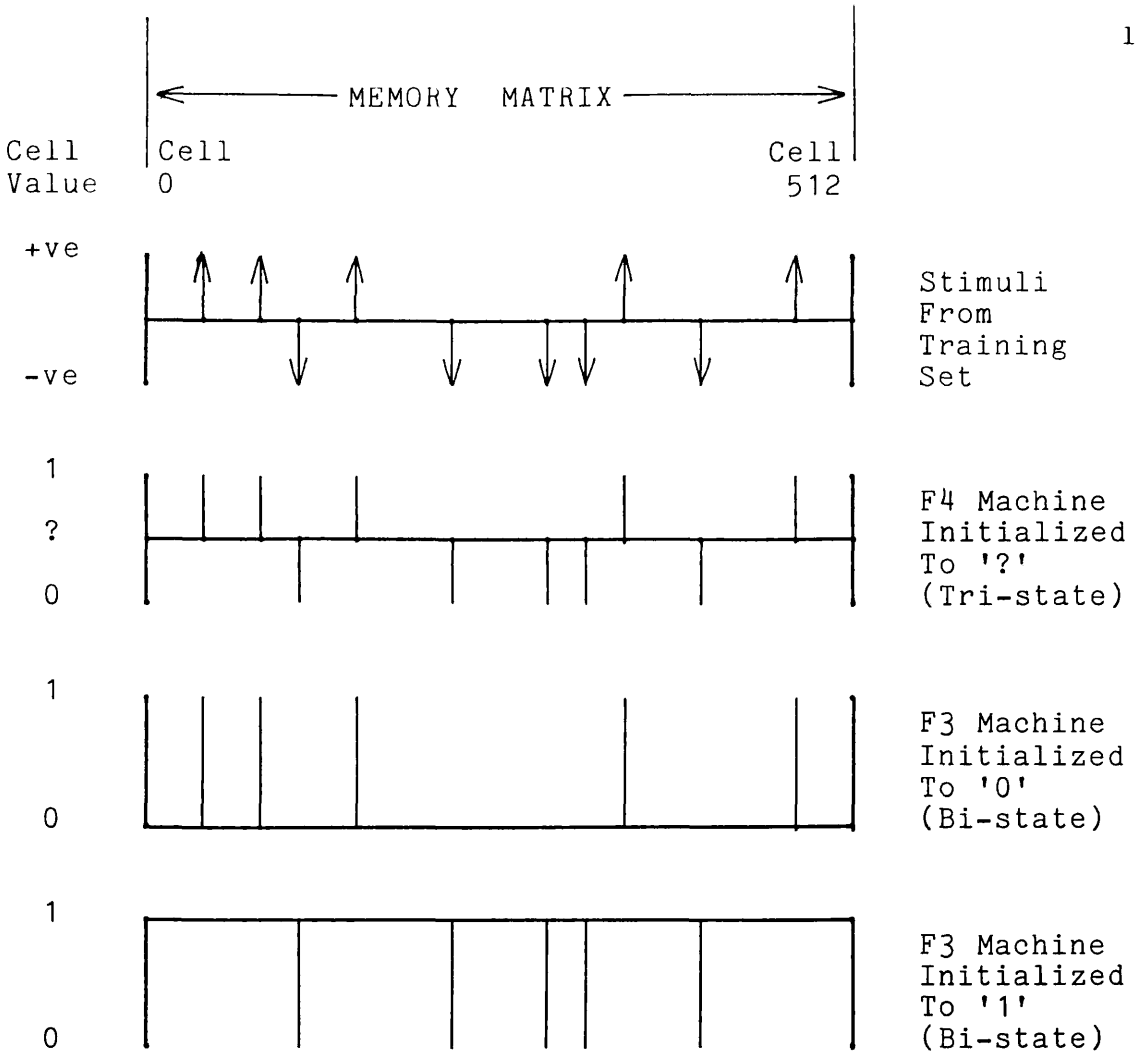


Fig 5.5 Effect of Training Stimuli on Memory Matrices

parallel mode, and the three pixel values are depicted as :

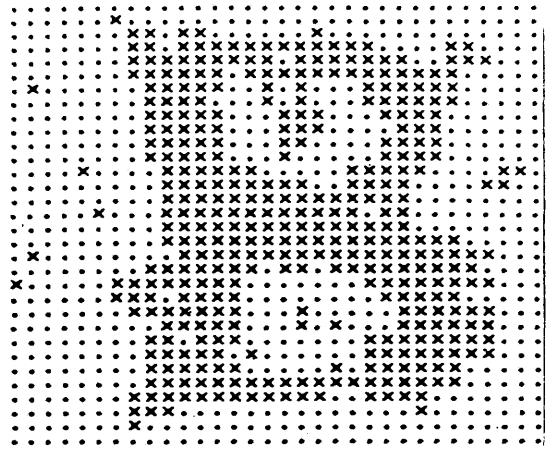
X	if cell value	=	1
?	if cell value	=	?
.	if cell value	=	0

(The training was shown in the top row of Fig 5.1).

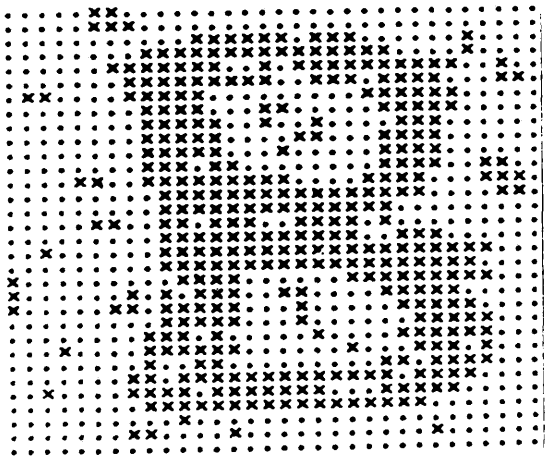
Results

The OPP generated by the F4 machine in the centre row can be seen to differentiate between the regions of the test patterns over which it can legitimately generalize. This

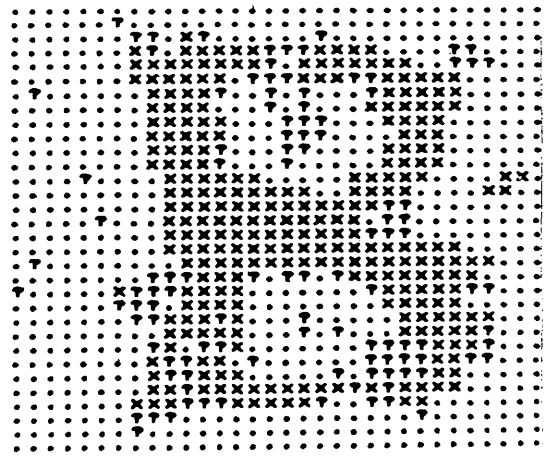
OPP F3 Bi-state m/c Init=1



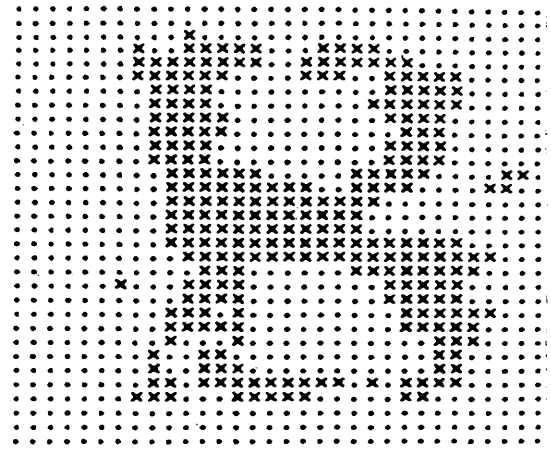
IPP All Machines



OPP F4 Tri-state m/c Init=?



OPP F3 Bi-state m/c Init=0



Operation : CLEAN

Fig 5.6 Comparison of Bi-state/Tri-state Machines' OPPs

gives an indication of how suitable and complete was the training received earlier.

Discussion

The F4 machine requires no more internal processing than the F3 machine. The only changes involve the storage and handling of potentially three-valued as opposed to two-valued variables. This involves a slight increase in storage by a factor ($\log_2 3/\log_2 2$).

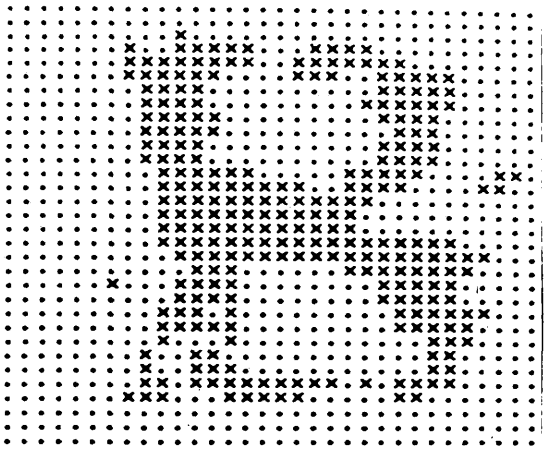
Testing on Four Tasks

A more complete test of the performance of the F4 machine was run. The range of picture processing tasks (clean, invert, thin, thicken) attempted in Experiment 6 with an F3 machine was repeated with the new variant.

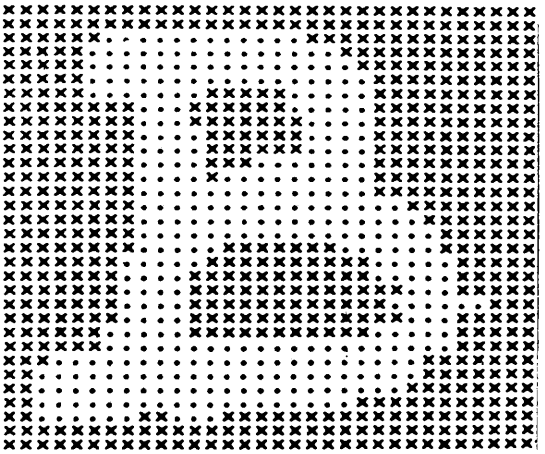
The training received in both cases was identical, and corresponds to the top rows of IPP+EXP patterns in Figs 5.1 to 5.4. The test IPPs and two OPPs (one from each machine) are shown for each of the four tasks in Figs 5.7 and 5.8.

The F4 machine can be seen to generate '?' pixels in regions where the training has been insufficient to allow the machine to generalize. Since neither machine is able to generate reliable information in these areas, more useful data is conveyed to the OPP field if this distinction between 'reliable' and 'unreliable' pixels is displayed. Thus, in the experiments that follow this tri-state memory matrix will be used predominantly as the new standard machine layout.

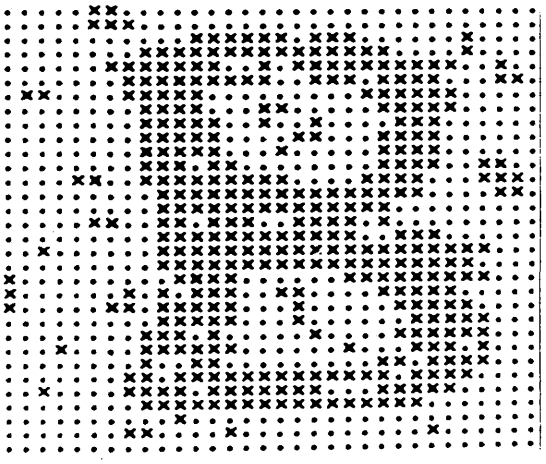
OPP Bi-State (Init=0) CLEAN



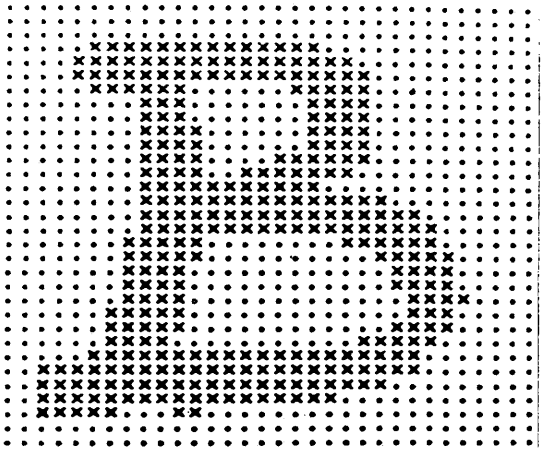
OPP Bi-State (Init=0) INVERT



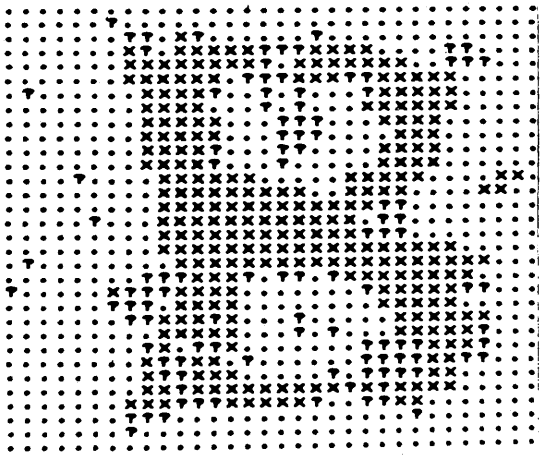
IPP Both Machines



IPP Both Machines



OPP Tri-State CLEAN



OPP Tri-State INVERT

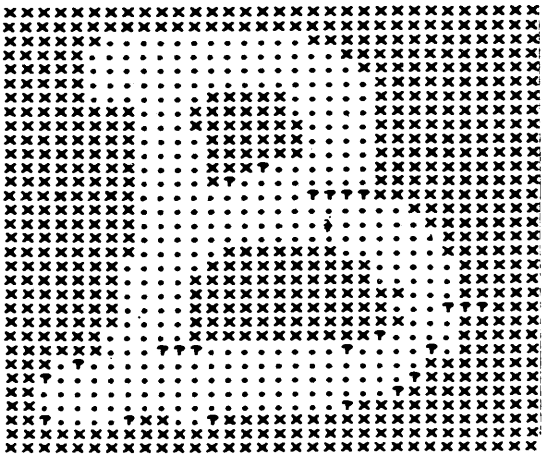
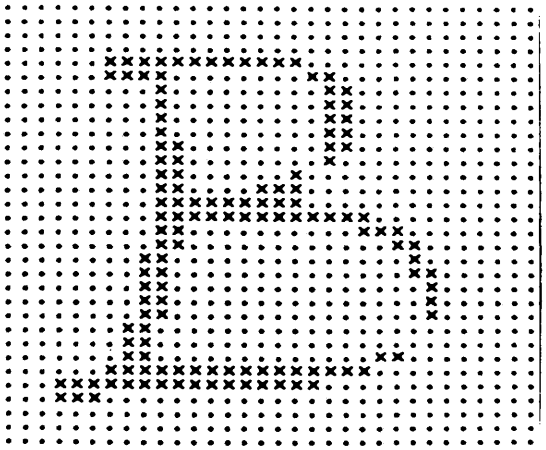
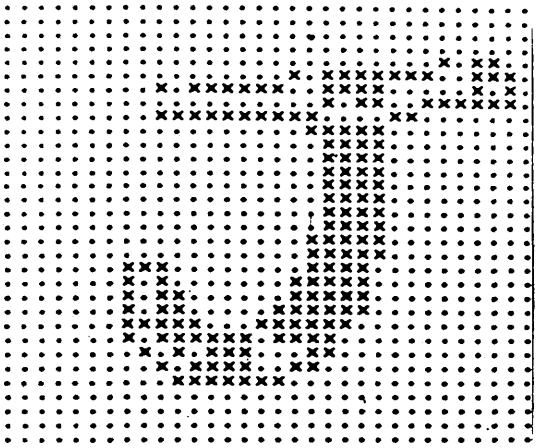


Fig 5.7 Experiment 7 : Bi/Tri-State MM : CLEAN,INVERT

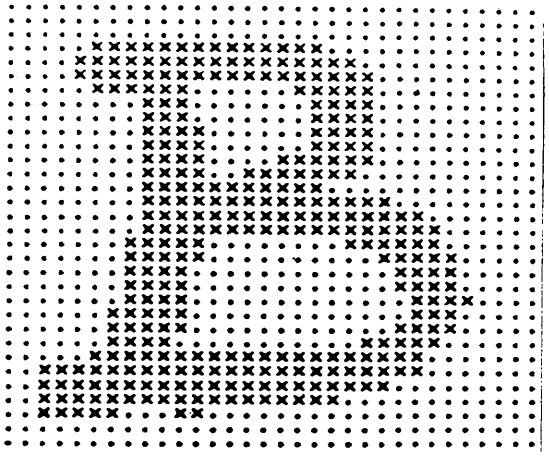
OPP Bi-State (Init=0) THIN



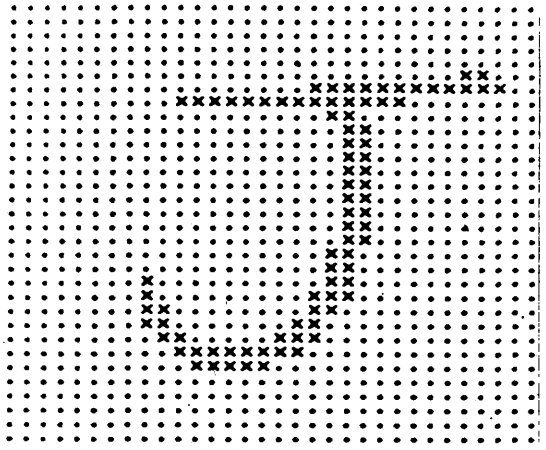
OPP Bi-State (Init=0) THICKEN



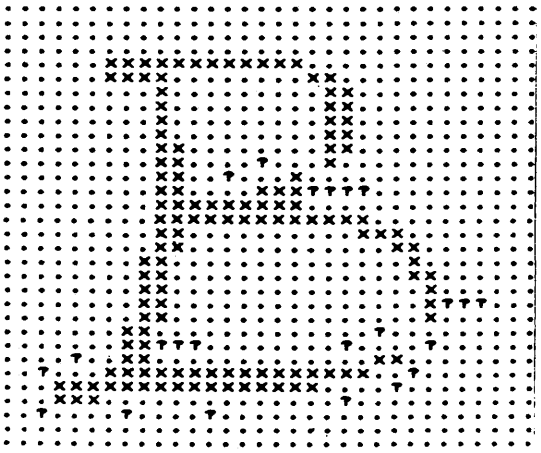
IPP Both Machines



IPP Both Machines



OPP Tri-State THIN



OPP Tri-State THICKEN

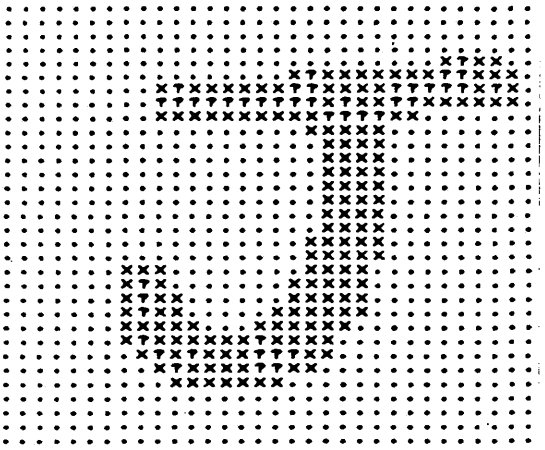


Fig 5.8 Experiment 7 : Bi/Tri-State MM : THIN,THICKEN

5.4 The Concept of a 'Trained Percentage'

The fact that a F4 machine uses tri-state memory matrix cells can be used to generate a measure of the amount of training received by the machine. This is primarily because cells that are unaffected by training stimuli are now distiguishable from those that have been set or cleared by such stimuli. Consequently, the number of cells remaining in the initial ('?') state after training can be compared with the number of cells that have been altered (to contain '0' or '1'). This ratio can be expressed as :

$$\frac{(\text{ number of trained cells })}{(\text{ total number of cells })} \times 100\ \% = \text{Percentage of Trained Cells}$$

and will be referred to as the 'trained percentage'. In a system of tri-state cells, it can be seen equivalent to :

$$\frac{(\text{ number of cells = 0 or 1 })}{(\text{ total number of cells })} \times 100\%$$

or :

$$\frac{(\text{ total number of cells }) - (\text{ number of cells = ? })}{(\text{ total number of cells })} \times 100\%$$

The F3 (bi-state) machine cannot be used to calculate simply this trained percentage, but the F4 machine can give it directly. Fig 5.9 shows these measurements being taken from these machines (F3 in two cases, and F4). The data used

Before Training :	% Cells set to 0	% Cells set to ?	% Cells set to 1
F3 Machine Init=1	0	-	100
F4 Machine Init=?	0	100	0
F3 Machine Init=0	100	-	0
After Training :			
F3 Machine Init=1	21.9	-	78.1
F4 Machine Init=?	21.9	64.4	13.7
F3 Machine Init=0	86.3	-	13.7
Key to Histograms			

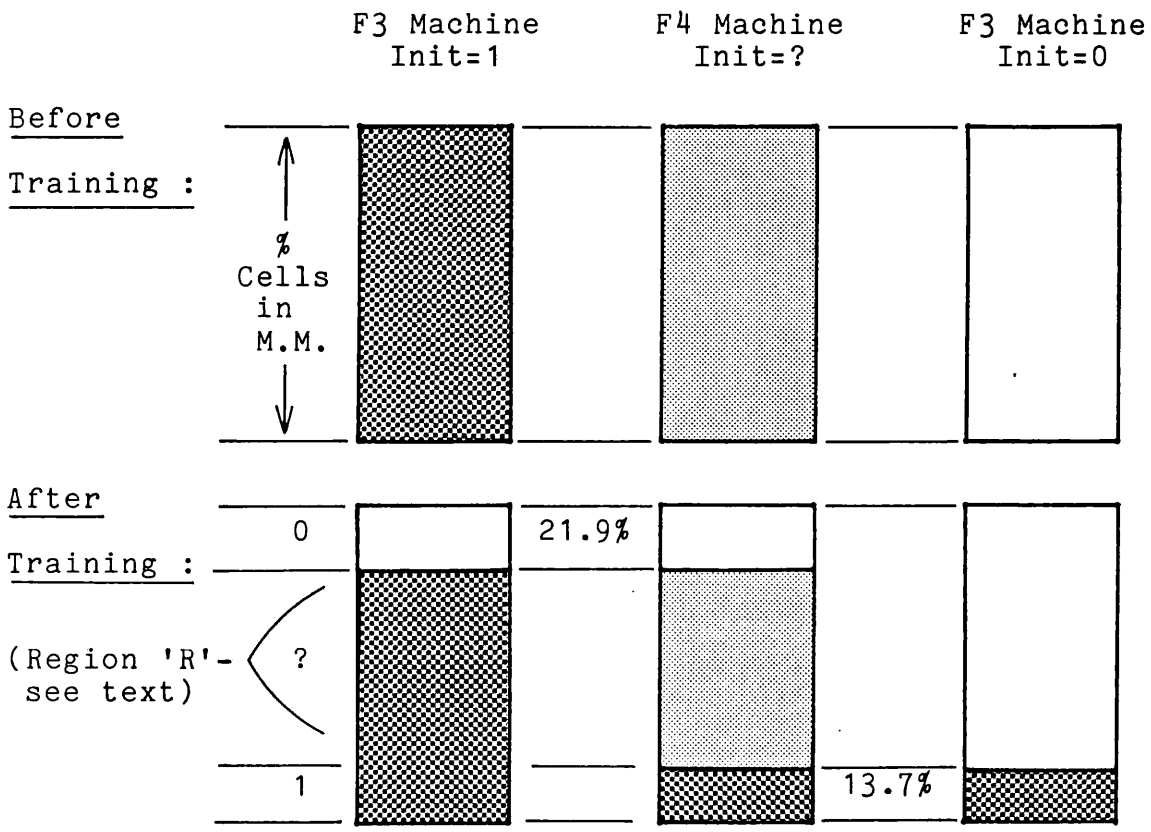


Fig 5.9 Percentage Totals of Memory Matrix Cells

in this figure were obtained by examination of the memory matrices after training in the first part of Experiment 7 (the pattern results having been shown in Fig 5.6).

Analysis

From Fig 5.9 it can be seen that the operation represented in Fig 5.5 is occurring - where there is a central band of residual cells in the machines which remain in the arbitrary initialisation state. (These are marked as 'Region R' in Fig 5.9.) If this state is distinguishable from the '0' and '1' states then the number of cells in this state can be measured directly. Thus the trained percentage can be most easily found from the F4 machine and is equal to :

$$(100 - 64.4) \% = 35.6 \%$$

This means that 35.6% of the cells were affected by the training set, which in this case was the IPP+EXP pair shown in the top row of Fig 5.1.

Prior to the use of this trained percentage value ('TP'), the only meaningful measure of the performance was a subjective evaluation of the resultant test OPPs. Now a more exact measure has been developed, although it must be noted that it only refers directly to the training quantity and only indirectly to the testing performance (ie. it reflects those cells whose value should have become either '0' or '1', but gives no indication which of these values is the more appropriate).

It should also be noted that this TP value can be a misleading figure of merit as it has a limiting value of 100%, and ultimately cannot be proportional to any

performance figure. A LPP machine with a TP of 100% ('fully trained') may still not give optimal picture processing performance. This only means that every cell has been accessed by the training set, but not that this training was necessarily correct. It will be shown later in Experiment 11 that while TP increases with a larger training set, again it is not a proportional increase. There is an asymptotic limit dependant on the quality of the training received.

Although these three factors : the quantity of training, TP value and the test performance are all related, and increase together, their relationship is neither linear nor readily quantifiable. However, relative changes in TP can can give clear indications of the effect of varying parameters in the LPP machine's operation. In this capacity, the TP value will be found a useful measure.

TP and the Memory Matrix Size

It has been noted that TP depends on the size of the memory matrix. This relationship may be more fully expanded. For an input window of w_i bits, there are 2^{w_i} cells in the memory matrix (assuming the cell addresses are formed by the simple stacking of binary pixels). When the machine is trained on a pattern of n bits, n accesses are made to the memory matrix. These will access any number (up to n) of different cells. The exact number depends on the variation in features in the training pattern. On a random data model of the input pattern, we have:

Number of training examples per pattern $\leqslant n$

Number of memory matrix locations $= 2^{w_i}$

thus: Addition to TP value per pattern $\leqslant \frac{n.100\%}{2^{w_i}}$

for 'N' pictures in a training set:

Total TP value after receiving N patterns $\leqslant$

$$\frac{N.n.100\%}{2^{w_i}} = \frac{\text{Total training received}}{\text{Memory Matrix Size}}$$

(The inequality simply illustrates that pictures generally contain a limited and repeated subset of all possible 2^{w_i} features.)

The TP value will remain constant for a variation in N , n or w if the expression

$$N.n \;/\; 2^{w_i}$$

remains constant. As an example, four extra binary pixels could be added to the w_i bit feature window. This would require a training picture with sixteen times the number of pixels, or alternatively sixteen times the number of training pictures to reach the same TP value.

While this analysis contains gross assumptions regarding the lack of redundancy in such training patterns, it illustrates the problems that would be encountered if very large windows were used. Enormous quantities of training would be required to maintain a significant TP value in the bigger memory matrix. The fact that the memory matrix grows exponentially with the window size results in two limiting factors on this arrangement of a LPP machine : the storage needed and the training requirements.

The Use of TP in Analysing Experimental Results

In this subsection, an analysis is made of how the TP varies with the picture processing task being learnt. The TPs for the memory matrices (trained to perform four separate tasks) in the latter half of Experiment 7 are given in Fig 5.10. Also shown are the distributions of cells set to the three possible states.

From these data, it can be seen that the TP value for the tri-state machine (F4) is more or less task independent - thin and thicken tasks have similar TP but widely differing numbers of cells set to alternately '0' or '1'. This means that the F4 machine is not muddled by pictures that are mostly '0's or mostly '1's, and TP can be a valid measure of training.

However, the exact value of TP in this machine depends only on the number of different 3x3 bit window features seen in the training IPP. In the case of the tasks invert and thin, the training IPP was the same. (See Figs 5.2 and 5.3 - the left patterns in the top rows are identical : a solid, thick letter A.) Consequently, these patterns accessed exactly the same memory matrix cells in both cases, even though the different EXPs then caused different data to be loaded into these cells. As a result, the TPs for these two tasks are identical, even though the number of cells equal to '0' and '1' differ, reflecting the different training. This highlights that TP is not, therefore, an ideal measure in all respects.

TASK	No. of Cells=0	No. of Cells=?	No. of Cells=1	No. of Trained Cells
Clean	112 (21.9%)	330 (64.4%)	70 (13.7%)	182 (35.6%)
Invert	32 (6.2%)	450 (87.9%)	30 (5.9%)	62 (12.1%)
Thin	61 (11.9%)	450 (87.9%)	1 (0.2%)	62 (12.1%)
Thicken	1 (0.2%)	436 (85.2%)	75 (14.6%)	76 (14.8%)

Histogram of Above Data : (Key as in Fig 5.9)

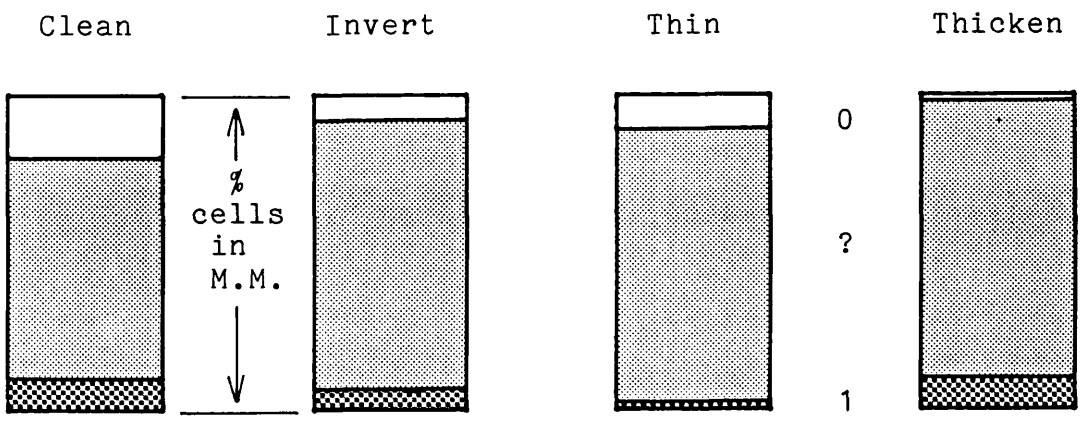


Fig 5.10 Distribution of Exp 7 F4 Machines Memory Matrix Contents

5.5 Experiment 8 : Augmenting Training by Window Symmetry Operations

It has been noted how the effectiveness of the training set can be measured by examining the TP value after training. In an effort to increase this TP value it was recognised that many of these feature cells correspond to effectively identical features.

For example, many tasks show no dependence on the feature direction in the pattern field. This results in independence from rotation and reflection of the input window in isotropic picture processing. The features shown in Fig 5.11 should all generate identical OPP pixels.

```
. . X     X . .     X . X     . . .
. . .     . . .     X . .     X . .
. X X     X X .     . . .     X . X

X X .     . X X     . . .     X . X
. . .     . . .     . . X     . . X
X . .     . . X     X . X     . . .
```

Fig 5.11 Example Features requiring the same OPP Pixel under any Combination of Rotation and Reflection

This indicates that more useful information could be extracted from the training set, if this principle of rotation and reflection of the feature window is utilised. A reduction in the memory matrix size would be an alternative as a number of cells are equivalent and hence redundant.

An 'F5' machine was devised to take advantage of such rotation and reflection symmetries. This is similar to the earlier F4 version, changed only in the function 'f_2' which generates the address from the window contents (see Fig 4.2). Each window extracted will be used to generate up to eight addresses and these will all be used to train the memory matrix repeatedly. These eight addresses will be generated by the rotation and reflection of the window in accordance with the layout shown in Fig 5.12.

```
P2  P3  P4      P8  P1::P2      P6  P7  P8      P4  P5  P6
::              ::                              
P1::P0  P5      P7  P0  P3      P5  P0::P1      P3  P0  P7
                                    ::          ::
P8  P7  P6      P6  P5  P4      P4  P3  P2      P2::P1  P8

P8  P7  P6      P2::P1  P8      P4  P3  P2      P6  P5  P4
                    ::                  ::
P1::P0  P5      P3  P0  P7      P5  P0::P1      P7  P0  P3
::                                              ::
P2  P3  P4      P4  P5  P6      P6  P7  P8      P8  P1::P2
```

Fig 5.12 Eight Possible Rotations and Reflections Of a 3x3 Pixel Window

The memory matrix cells corresponding to these eight features will all be trained by the centre point value derived from the EXP pattern. Some of these permutations will generate identical addresses, dependent on the symmetry of the feature involved. (Some cells may be trained more than once with the same data, but the above method is used as it saves processing and rationalises the operation for any input window, whatever its symmetry. In any case, this cannot give misleading results, as once a cell is set,

setting it again has no additional effect with this machine. Another important point is that the overhead of checking to reduce the effort of such repeated training, might well itself consume more effort than it saves.)

The retention of all 512 cells also simplifies the test procedure, which is carried out exactly as in the F4 machine. The test IPP windows address the relevant cells (which may now have been trained by equivalent, rather than identical features) and the contents are output as the new OPP pixels.

Experiment

The same four tasks used for training the F4 machine in the previous experiment were used again here for comparison. (This training data was shown at the top of Figs 5.1 to 5.4). The test IPPs and OPPs from both the F4 and F5 machines are compared in Figs 5.13 and 5.14. The TP values are tabulated in Fig 5.15.

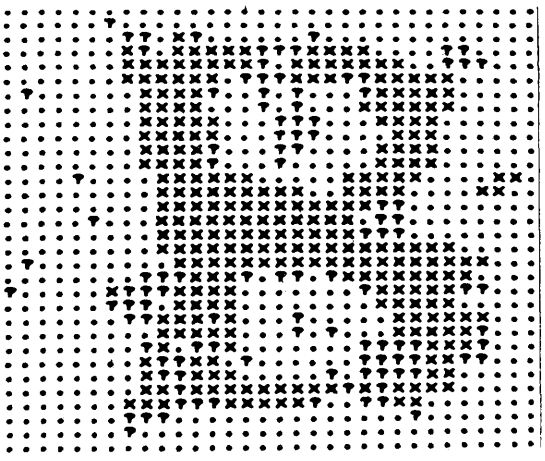
Discussion

Examination of the data in Fig 5.15 reveals the expected increase in TP as a result of rotating and reflecting the input window from the training pattern. This shows in testing as an improvement in the OPPs from the F5 machine which have generally fewer '?' pixels, indicating improved generalisation ability. This is most notable in the case of the cleaning task, where there is a large reduction in the undefined pixels - the TP value changes from 35.6% to 65.8% .

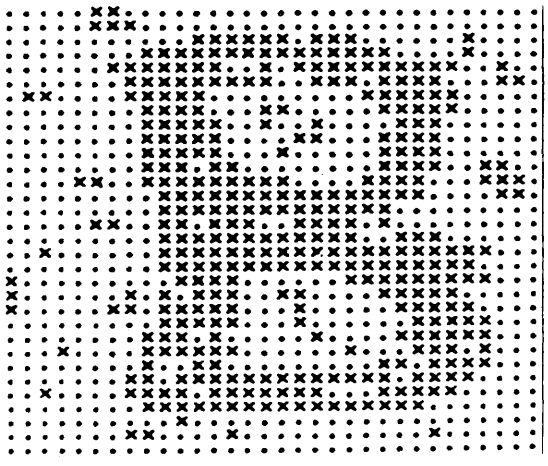
Also shown in Fig 5.15 are the average TPs for the two

CLEAN

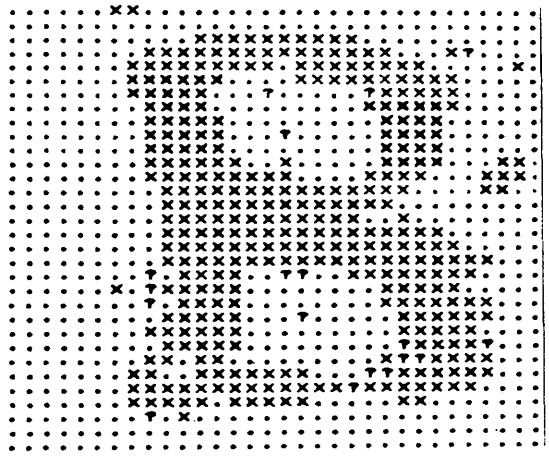
OPP F4 m/c NO (r+r)



IPP Both Machines

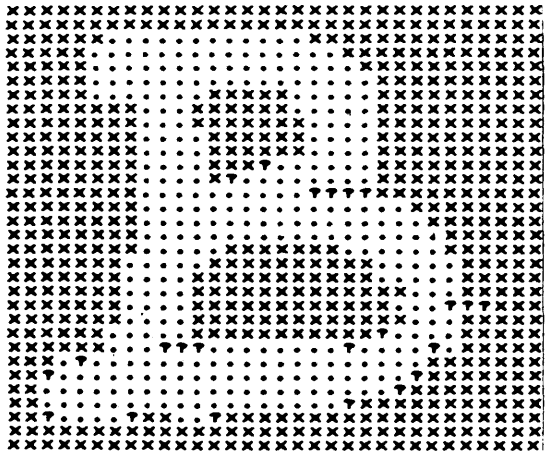


OPP F5 m/c With (r+r)

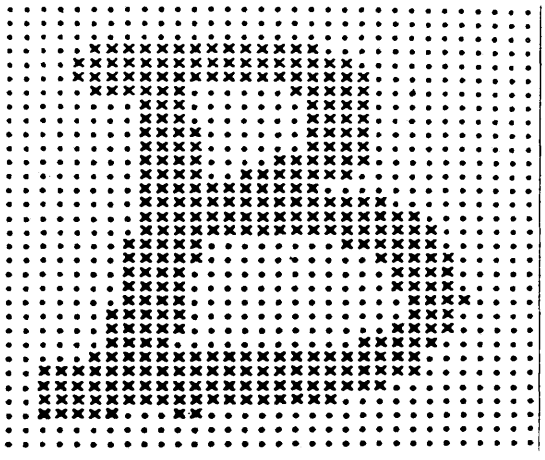


INVERT

OPP F4 m/c NO (r+r)



IPP Both Machines



OPP F5 m/c With (r+r)

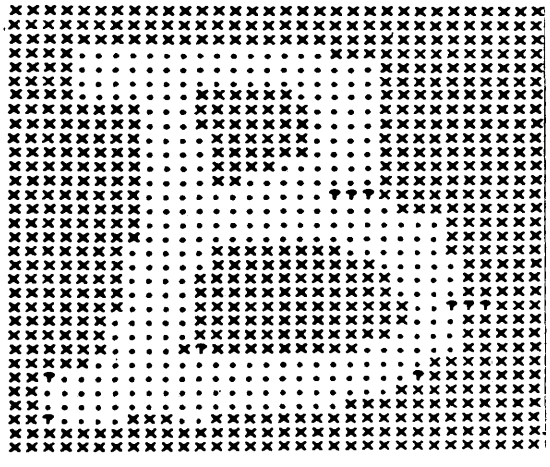
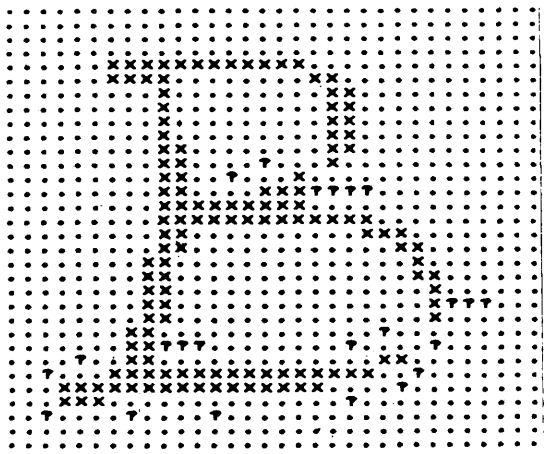


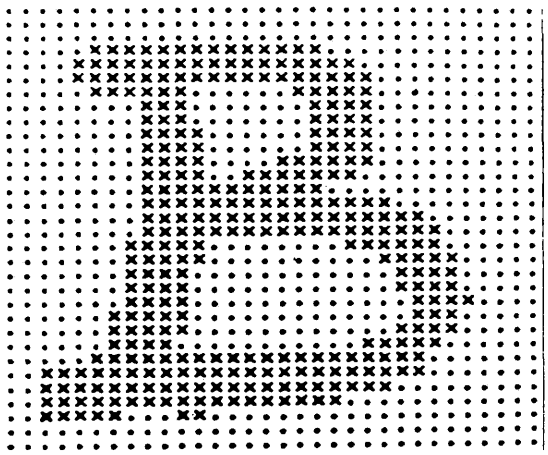
Fig 5.13 Experiment 8 : Test Results Comparison
With/Without (r+r) : Tasks : CLEAN , INVERT

THIN

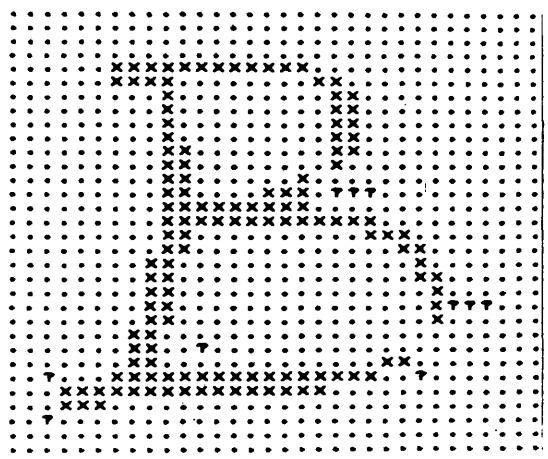
OPP F4 m/c NO (r+r)



IPP Both Machines

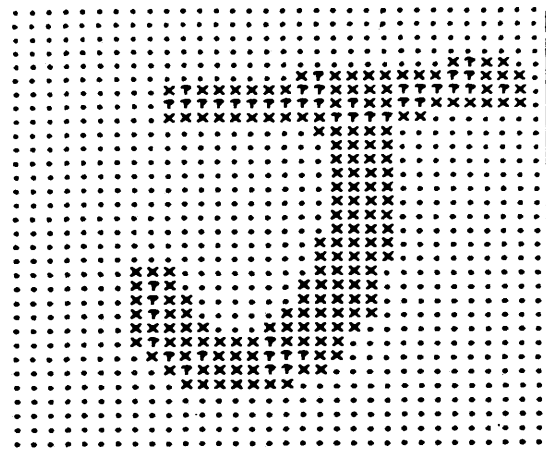


OPP F5 m/c With (r+r)

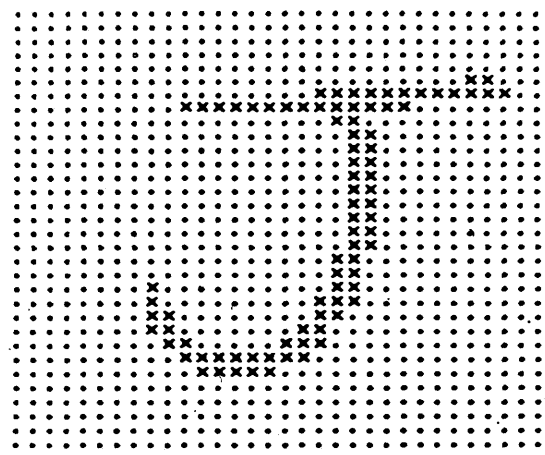


THICKEN

OPP F4 m/c NO (r+r)



IPP Both Machines



OPP F5 m/c With (r+r)

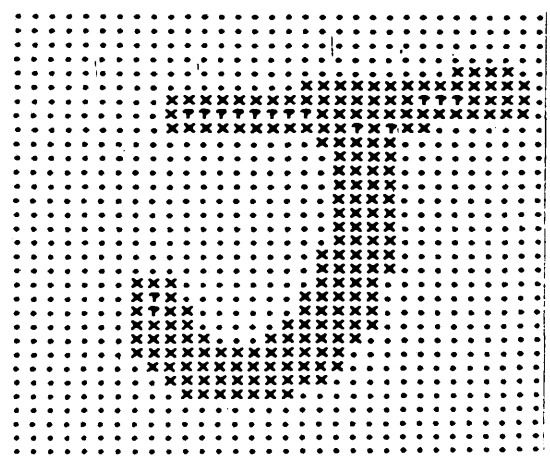


Fig 5.14 Experiment 8 : Test Results Comparison
With/Without (r+r) : Tasks : THIN , THICKEN

	F4 m/c NO (r+r)				F5 m/c with (r+r)			
TASK	Cells % = 0	Cells % = ?	Cells % = 1	T P (0+1)	Cells % = 0	Cells % = ?	Cells % = 1	T P (0+1)
Clean	21.9	64.4	13.7	35.6	35.7	34.2	30.1	65.8
Invert	6.2	87.9	5.9	12.1	7.2	84.0	8.8	16.0
Thin	11.9	87.9	0.2	12.1	15.8	84.0	0.2	16.0
Thicken	0.2	85.2	14.6	14.8	0.2	70.9	28.9	29.1
	Average TP F4 m/c =			18.6	Average TP F5 m/c =			31.7

Histograms of Above Data : (Key as in Fig 5.9)

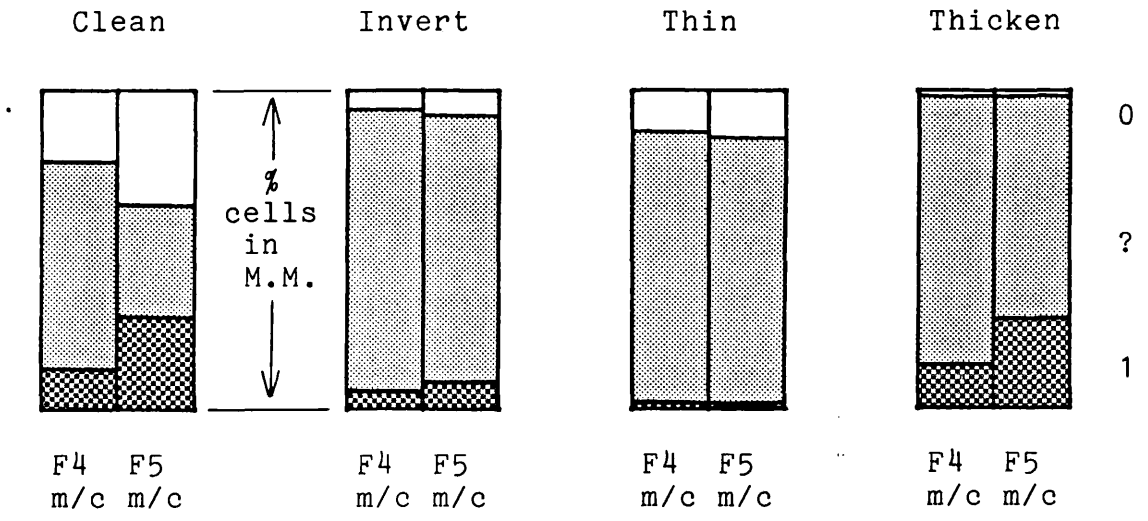


Fig 5.15 Comparison of F4 and F5 Machines' Memory Matrix Contents : With/Without (r+r)

cases of machines - with and without rotation and reflection. These are respectively 31.7% and 18.6% and represent a 1.7 fold increase in TP as a result of using this technique.

There is an important value that can be derived from the theory to predict this effective increase in training quantitively. It refers to the change in number of different possible 3x3 window features with and without rotation and reflection. Consequently, this factor of 1.7 will be compared later (Section 5.7) with this maximum possible theoretical increase (viz. 512/102).

5.6 Experiment 9 : The Effect of Scan Direction on an Anisotropic Picture Processing Task

The above use of symmetry, which extracts more information from the training set, relies on an isotropic picture processing task for coherent results. All four tasks used above were of this type and exhibited an increase in TP on the adoption of window rotations and reflections. However, the use of an anisotropic task (such as 'shift laterally') illustrates how the machine's output depends on the following :

1 the relationship between the direction of movement of the scanning window in training and the direction defined by the anisotropic task (eg. shift direction),

2 the use or not of rotation and reflection of the input window giving respectively incoherent or coherent results.

The combined investigation of these two effects gives four combinations of outputs to be examined experimentally. These are :

1	Scan in a given direction (eg. against the direction of shift)	-	use of (r+r) (F5 machine)
2	Scan in a given direction	-	NO use of (r+r) (F4 machine)
3	Scan in opposite direction (with the shift direction)	-	use of (r+r) (F5 machine)
4	Scan in opposite direction	-	NO use of (r+r) (F4 machine)

These four cases will be attempted below.

To facilitate execution of this experiment an alternative, but equivalent, method of effectively reversing the direction of scan will be used. Rather than physically reversing the scan direction - which has so far been arbitrarily set as left to right, top to bottom - the direction of the shift will be reversed. That is, cases 3 and 4 above will be run with a shift task in the opposite direction (viz. towards bottom right) to that used in cases 1 and 2 (towards top left). This method of effectively reversing the direction of scan when using an anisotropic picture processing task is used because it is simple to implement - requiring only the interchange of the IPP and EXP pair during training.

Experiment

The four cases specified above were run, using training and test characters from the same stock of hand-drawn,

binary 32^2 patterns used in the previous experiment. These are illustrated in Figs 5.16 and 5.17.

Results

The resultant OPPs generated in these four cases can all be seen to be quite different and will be briefly described here.

Case 1 - Scan in opposite direction to shift, F5 machine

This results in very few pixels left in OPP 1, only a few isolated points set to '1' or '?'. The object is essentially removed entirely from the field.

Case 2 - Scan in opposite direction to shift, F4 machine

The OPP 2 generated is a shifted version (towards top left) of the test IPP as required. However, some breaks in the limbs can be seen at junctions.

Case 3 - Scan in same direction as shift, F5 machine

This results in a generally unshifted and 'ragged' OPP 3 with many holes and spurs over the entire pattern in all directions.

Case 4 - Scan in same direction as shift, F4 machine

The OPP 4 is a correctly shifted version of IPP, towards the bottom right.

Discussion

The training of a F4 machine on an anisotropic task such as shift is relatively insensitive to the direction of shift compared to that of scan. That is, cases 2 and 4 above performed coherently as trained and shifted the test IPP in the required directions.

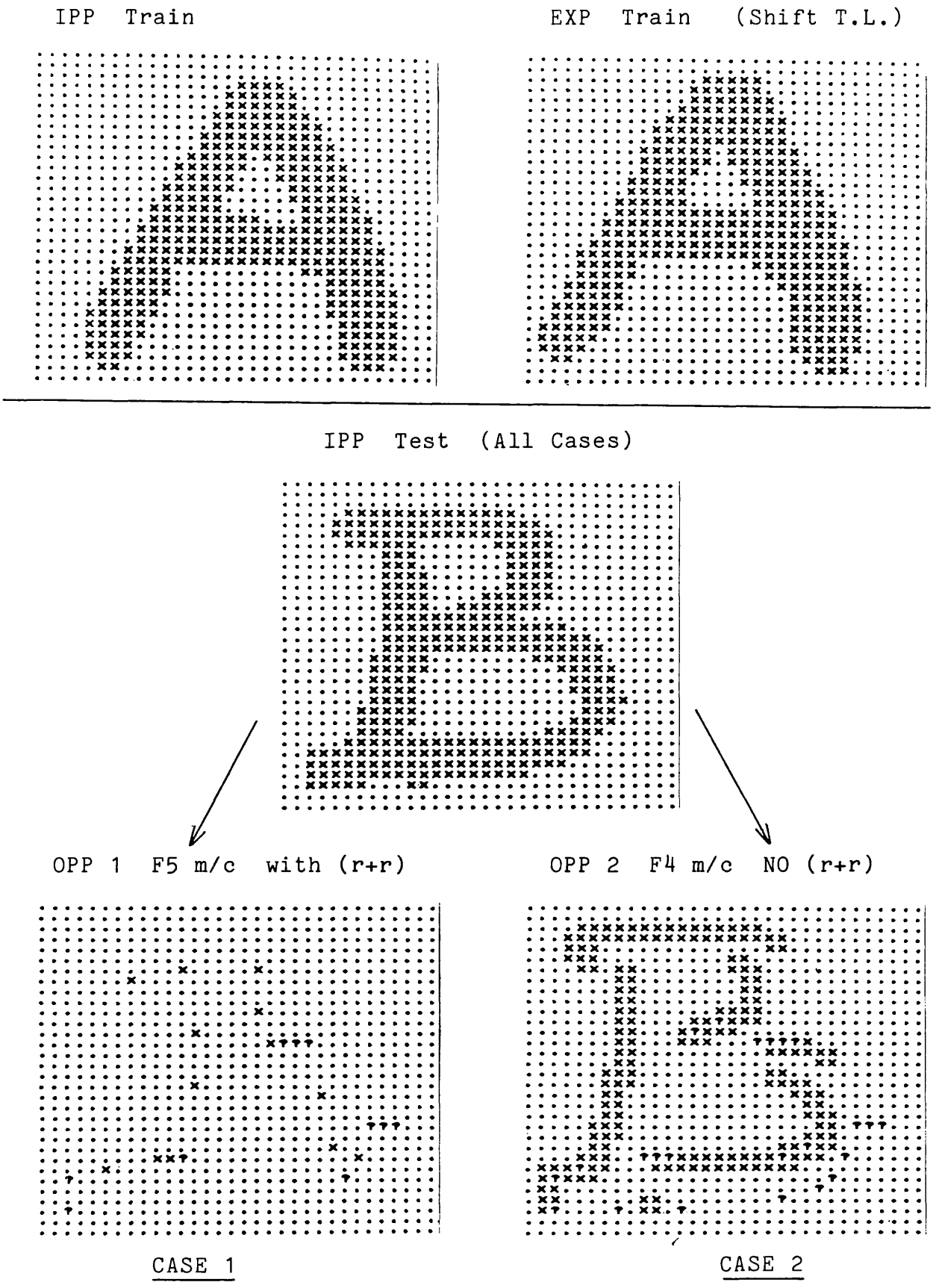
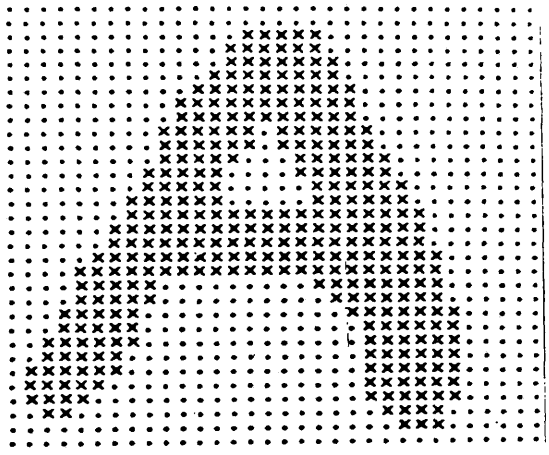
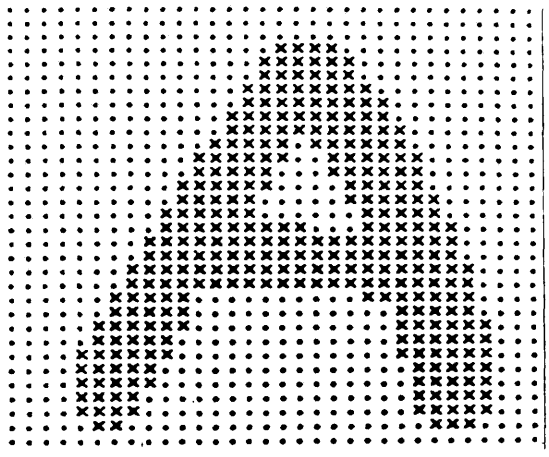


Fig 5.16 Experiment 9 : Cases 1 and 2 : SHIFT Top Left

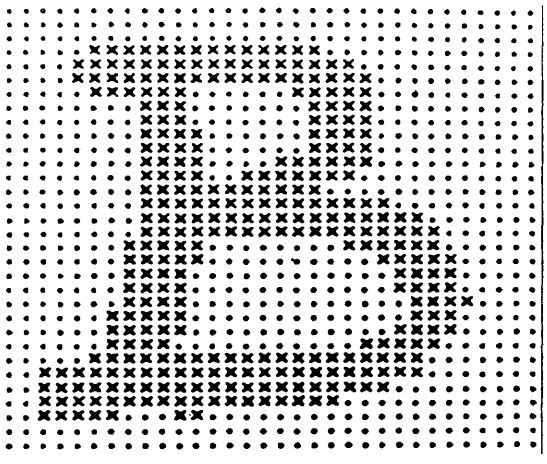
IPP Train



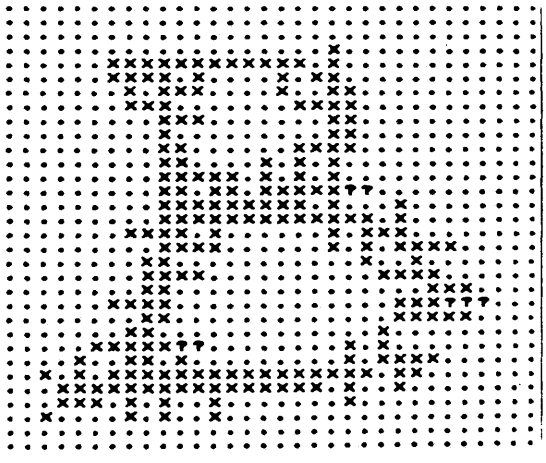
EXP Train (Shift B.R.)



IPP Test (All Cases)

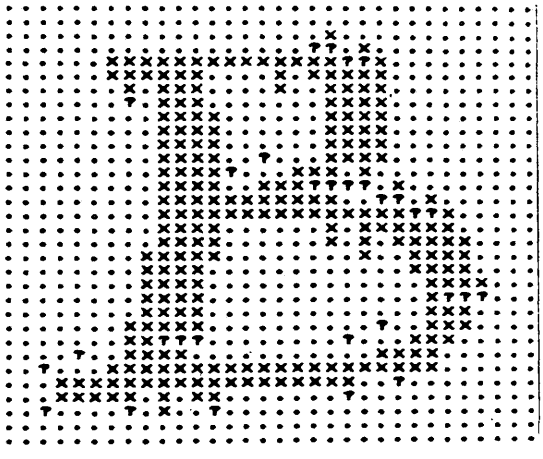


OPP 3 F5 m/c with (r+r)



CASE 3

OPP 4 F4 m/c NO (r+r)



CASE 4

Fig 5.17 Experiment 9 : Cases 3 and 4 : SHIFT Bottom Right

However, in cases 1 and 3 - the F5 machine - the attempt at utilizing rotation and reflection has resulted in widely differing results dependent on the shift direction. This behaviour can be explained by the following considerations.

In the case of the shift direction being in the opposite direction to the scan direction (Case 1), the effect of the training set on the memory matrix must be examined. It must be remembered that this is a temporal process as patterns are scanned sequentially. The memory matrix is gradually built up by successive applications of the input and example window extractors (f_1 and f_4 of Fig 4.2) to the training pair. Consequently, the stimuli received towards the end of the scan will have the more recent effect than those at the beginning. Normally this is unimportant as in the case of isotropic tasks, the training would be consistent - <u>wherever</u> it originated in the training pair, and hence - <u>whenever</u> it was received in the scan period. However, in the case of an anisotropic task ('shift top left') the stimuli from the top of the picture (beginning of scan) is different from that at the bottom of the picture (end of scan). The stimuli could be formalized in general terms as follows :

(a) stimuli from the top of the picture cause the machine to generate '1' pixels,

(b) stimuli from the bottom of the picture cause the machine to generate '0' pixels.

The rotation and reflection of the input window removes the possiblity of dependence on orientation of the features

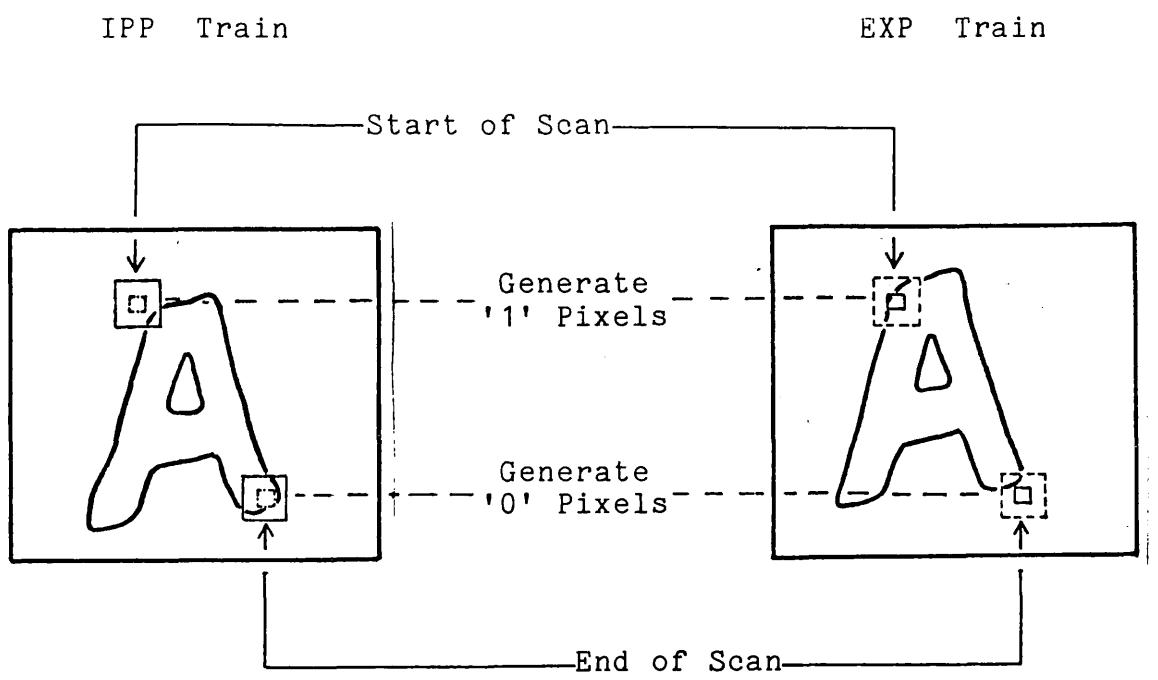
and hence no dependence on left, right, upper or lower edges can remain. The stimuli are reduced simply to generate '0' or '1' pixels from symmetrical features. That is, the training is inconsistent and generates a state of dynamic equilibrium in the memory matrix, where the contents at any point are indeterminate.

This is illustrated in Fig 5.18a where the training pair IPP+EXP are shown with sample windows extracted towards both the beginning and end of the scan period. The stimuli can be seen to vary from 'generate 1 pixels' to 'generate 0 pixels' as the scan proceeds in this case. Consequently, as the later stimuli predominate, the test run in the experiments just performed reveals a memory matrix trained to generate predominantly '0' pixels. This can be seen in the lower left pattern of Fig 5.16 (Case 1).

The reverse is true in the case of the opposite shift direction (Case 3). As shown in Fig 5.18b the training stimuli received later in the scan result in the generation of '1' pixels, resulting in a OPP (lower left of Fig 5.17) that is generated by a memory matrix trained in this manner. It should be noted that although this OPP is generally full of spurs and holes, it is not shifted in any particular direction. (The 'noisy' result is due to this inconsistent training received from the beginning and end of the characters, for they are not symmetrical.)

This experiment has demonstrated clearly that these machines rely most heavily on the most recent information received in training. This means, in the case of a sequential scan, the 'end' of a picture will predominate. If this differs from the transformation seen at the beginning

(a) SHIFT Towards Top Left



(b) SHIFT Towards Bottom Right

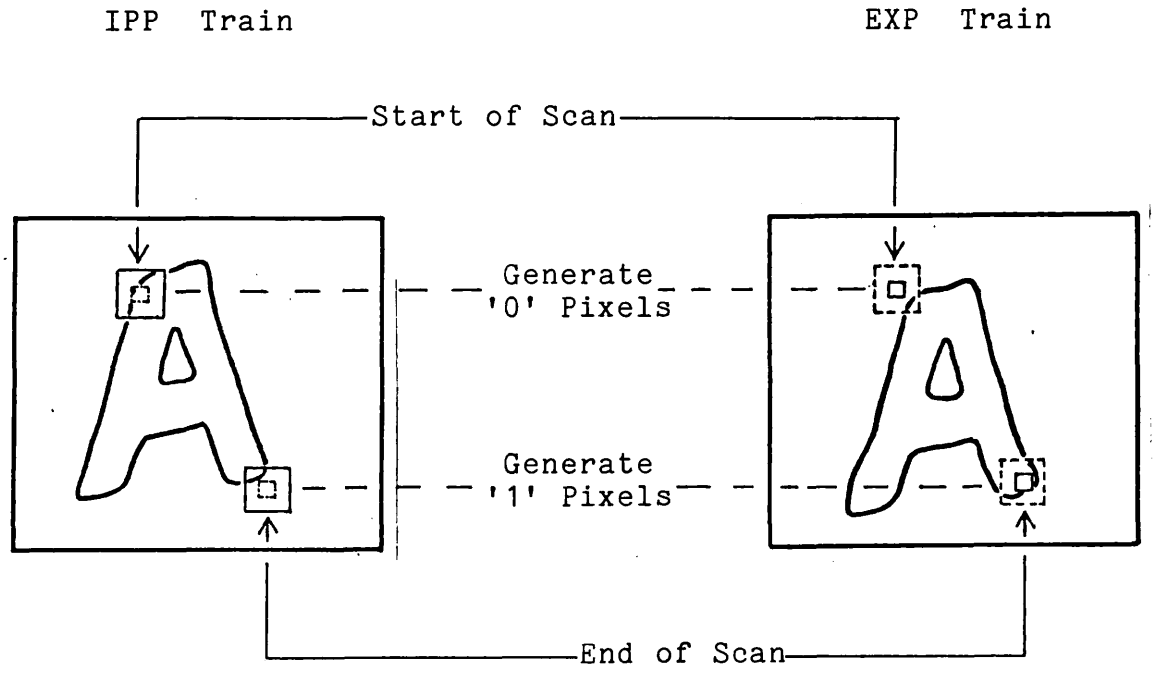


Fig 5.18 Training Stimuli Received at Start and End of Scans for Two Shift Operations

of the picture, the machine is subject to 'neurosis' - it cannot resolve the inconsistency. (This 'neurotic' aspect of behaviour is covered in some depth in Experiments 16 and 17.)

5.7 Experiment 10 : 5-bit Window and the Direct Examination of the Memory Matrix

It has been mentioned in earlier sections that the memory matrix can be examined directly after training as an aid to understanding how the LPP machine operates. This direct examination of the memory matrix cells' contents has not yet been implemented. Only the integrated totals of numbers of cells set equal to particular values have been examined so far, producing the type of data seen earlier in Figs 5.9 and 5.10.

However, the interrogation of the LPP simulator will supply these cell data contents directly, albeit in a rather unwieldy form - a table of 512 tri-state variables. In order to appreciate these data, it will be helpful to temporarily change the format of the LPP machine. This will result in less bulk data to be examined, and will facilitate later understanding of a larger (512 cell) memory matrix.

The 5-bit Window Format

The 9-bit window used so far will now temporarily be replaced by a 5-bit window, consisting of a centre point and its immediate four-connected neighbours on a rectangular lattice. This is transformed by the address calculator (f_2 of Fig 4.2) into a 5-bit binary address in the range 0-31.

The remainder of this new machine ('F6' Format) is identical to the F4 (9-bit window) machine. This new window format is illustrated below in Fig 5.19.

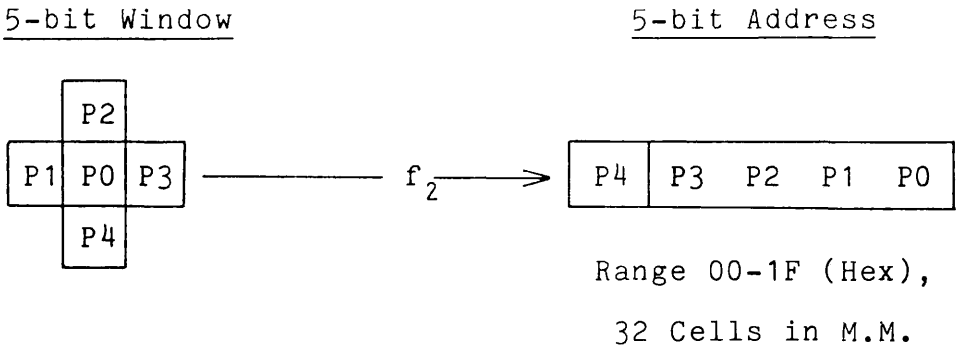


Fig 5.19 5-bit Window Format used by F6 LPP Machine

Experiment

This F6 machine was trained on the same data as the F4 machine in Experiment 7 : the four tasks : clean, invert, thin, thicken. These training patterns were illustrated in Figs 5.1 to 5.4. A listing of the addresses, corresponding 5-bit features and the resultant cell data contents are shown below in Figs 5.20 and 5.21. This is given for each of the 32 cells in the memory matrix and also for each of the four tasks involved.

This listing illustrates how the LPP machine has been trained to react to each of the 32 possible 5-bit input features. For each of the four tasks, the output pixel that the machine would generate in testing is shown in the corresponding column. It may also be seen how TP values are generated, and their relevance to training of the memory matrix. These TP values for each of the four tasks are shown

M.M. Addresses, 5-bit Features and Cell Contents for the Four Tasks : Clean, Invert, Thin, Thicken.

CELLS 00 - 15

M.M. Address		Feature	Cell Contents trained to :			
Dec	Binary		CLEAN	INVERT	THIN	THICKEN
00	00000	. . . . .	.	X	.	.
01	00001	. . X . .	.	?	?	?
02	00010	X	.	X	.	X
03	00011	X . X . .	X	?	?	X
04	00100	. . . X .	.	X	.	X
05	00101	. . X X .	.	?	?	X
06	00110	X . . X .	.	.	.	X
07	00111	X . X X .	.	.	.	X
08	01000	 X	.	X	.	X
09	01001	. . X . X	.	?	?	?
10	01010	X . . . X	.	?	?	?
11	01011	X . X . X	X	?	?	X
12	01100	. . . X X	X	X	.	X
13	01101	. . X X X	X	.	.	X
14	01110	X . . X X	X	X	.	X
15	01111	X . X X X	X	.	.	X

(0 = . ? = ? 1 = X)

Fig 5.20 Memory Matrix Listing for F6 Machine - Part 1

M.M. Addresses, 5-bit Features and Cell Contents for the Four Tasks : Clean, Invert, Thin, Thicken.

CELLS 16 - 31

M.M. Address		Feature	Cell Contents trained to :			
Dec	Binary		CLEAN	INVERT	THIN	THICKEN
16	10000	 X	.	.	.	X
17	10001	. X . . X	.	?	?	X
18	10010	X . . . X	.	X	.	X
19	10011	X X . . X	.	.	.	X
20	10100	. . X . . X	.	?	?	?
21	10101	. X X . . X	X	?	?	?
22	10110	X . X . . X	X	?	?	?
23	10111	X X X . . X	X	.	.	X
24	11000	. . . X X	.	X	.	X
25	11001	. X . X X	.	X	.	X
26	11010	X . . X X	X	?	?	?
27	11011	X X . X X	X	.	.	X
28	11100	. . X X X	X	?	?	?
29	11101	. X X X X	X	.	.	X
30	11110	X . X X X	X	?	?	?
31	11111	X X X X X	X	.	.	X

(0 = . ? = ? 1 = X)

Fig 5.21 Memory Matrix Listing for F6 Machine - Part 2

below in Fig 5.22, with data taken from Figs 5.20 and 5.21 in accordance with the formulae in Section 5.4.

For the sake of completeness, the test OPPs actually generated by this 5-bit F6 machine are shown in Figs 5.23 and 5.24, compared with the OPPs generated by the 9-bit F4 machine in Experiment 7. These two machines have received the same training and test inputs, and consequently the outputs differ only as a result of the window size being reduced. It is interesting to note that in the case of the 'thin' task the training appears too coarse for a smaller 5-bit window, yet the other tasks show reasonable results from both machines. (It is to be remembered that these results were obtained with a memory matrix of only 32 tri-state variables.)

With the above data, it can be seen how the F6 machine reacted to its training, and consequently generated the OPPs shown in Figs 5.23 and 5.24. An examination of each pixel in the test IPPs, together with its four neighbours, will indicate (from the look-up table of Figs 5.20 and 5.21) the corresponding OPP pixel.

For the 9-bit window machine, the equivalent look-up table is, of course, sixteen times larger and hence a direct examination of the full memory matrix would be excessively tedious. Methods for effectively reducing the number of cells to be examined are thus desirable.

Reduction in Memory Matrix size by Rotation and Reflection

Neither the F4 or F6 machines used above made use of rotation or reflection of the input window. However, this has already been achieved with the F5 machine. In addition to an effective increase in the training to be gained from a

F6 m/c Tri-state M.M.				
TASK	No. of Cells=0	No. of Cells=?	No. of Cells=1	No. of Trained Cells
Clean	17 (53%)	0 (0%)	15 (47%)	32 (100%)
Invert	10 (31%)	13 (41%)	9 (28%)	19 (59%)
Thin	19 (59%)	13 (41%)	0 (0%)	19 (59%)
Thicken	1 (3%)	9 (28%)	22 (69%)	23 (72%)

Histogram of Above Data : (Key as in Fig 5.9)

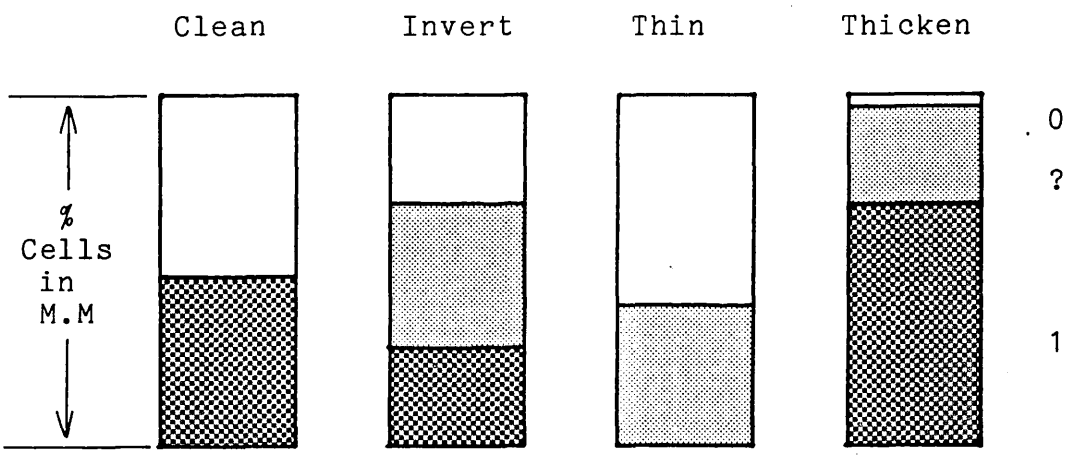
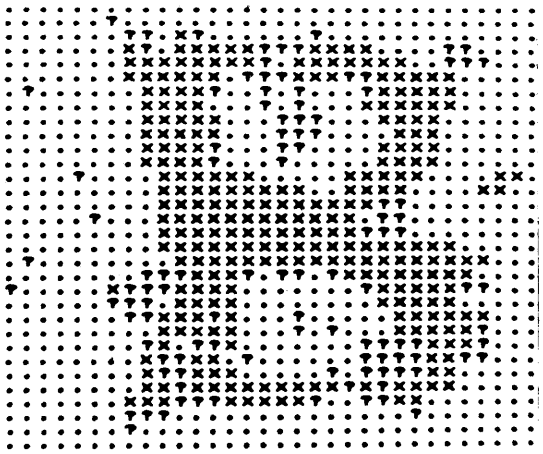


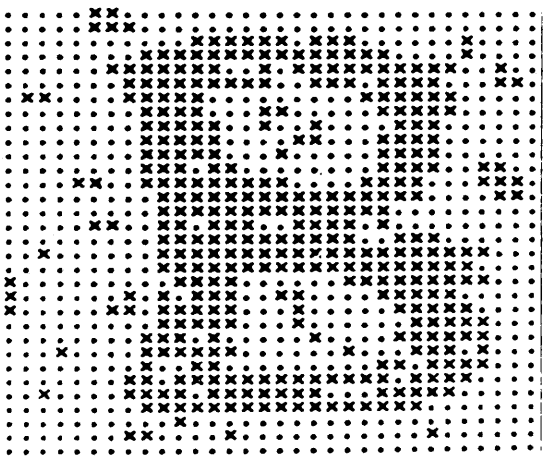
Fig 5.22 Distribution of M.M Contents of F6 Machine As Trained in Experiment 10

CLEAN

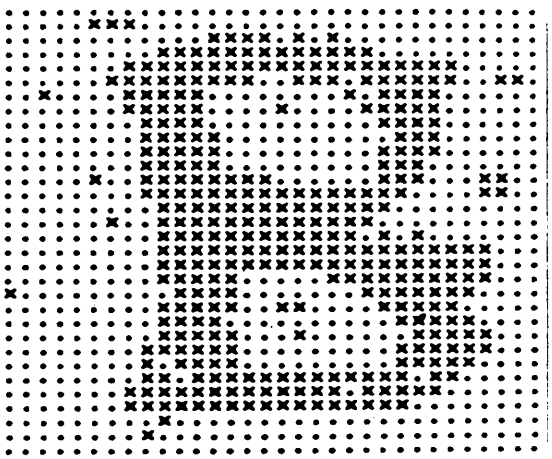
OPP F4 m/c 9-bit Window



IPP Both Machines

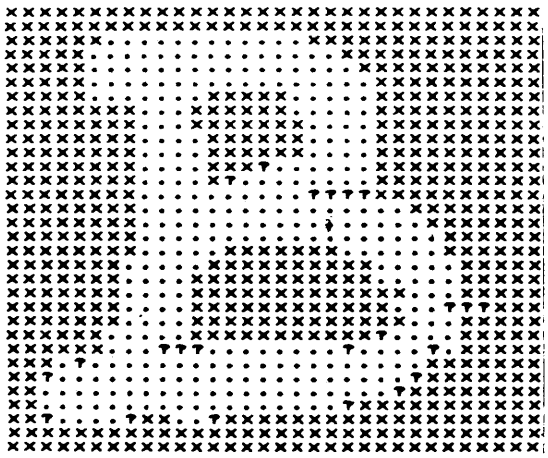


OPP F6 m/c 5-bit Window

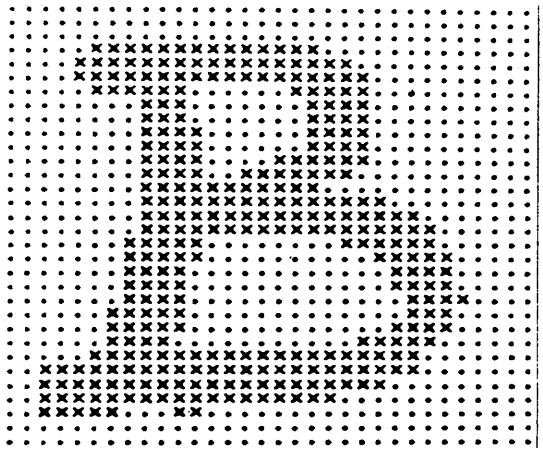


INVERT

OPP F4 m/c 9-bit Window



IPP Both Machines



OPP F6 m/c 5-bit Window

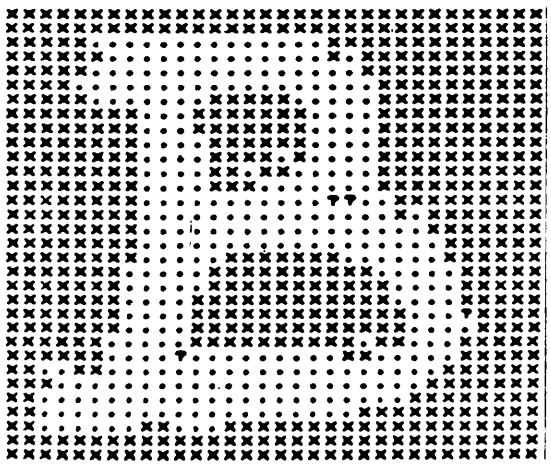
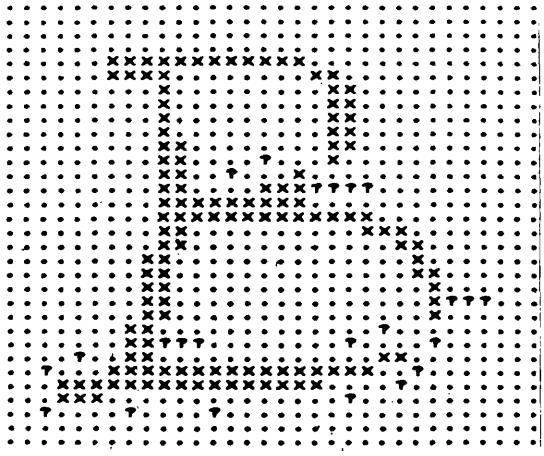


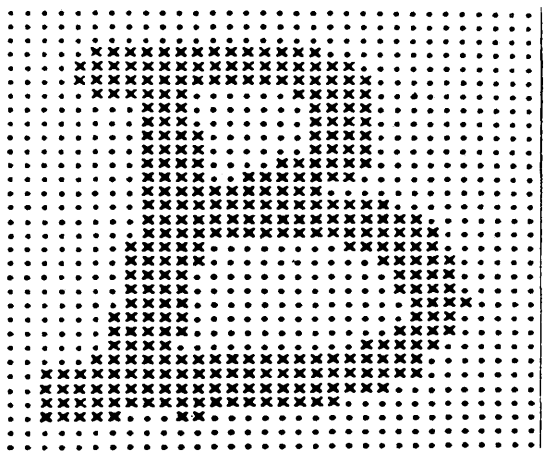
Fig 5.23 Experiment 10 : Tests Results : F4/F6 m/cs
Tasks : CLEAN , INVERT

THIN

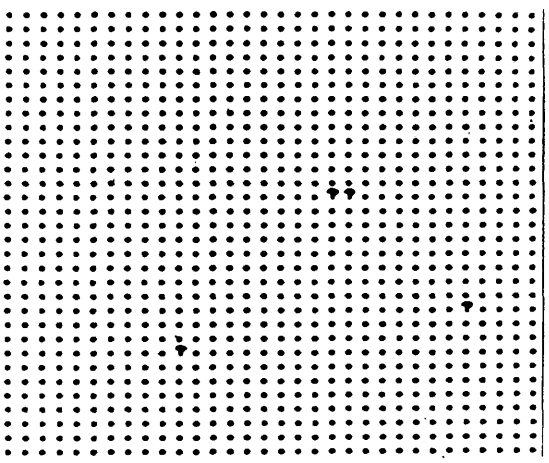
OPP F4 m/c 9-bit Window



IPP Both Machines

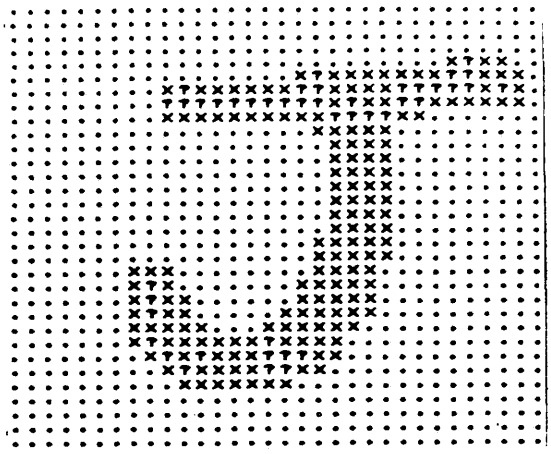


OPP F6 m/c 5-bit Window

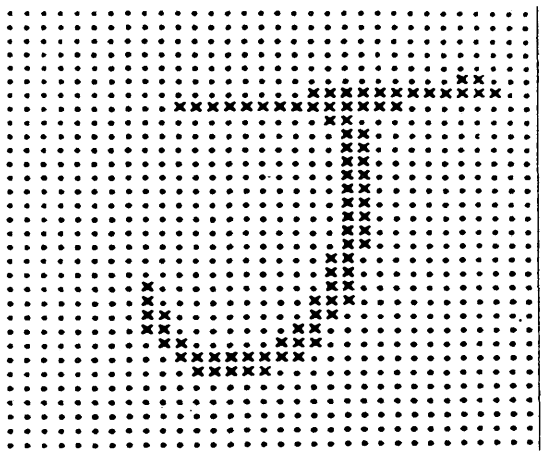


THICKEN

OPP F4 m/c 9-bit Window



IPP Both Machines



OPP F6 m/c 5-bit Window

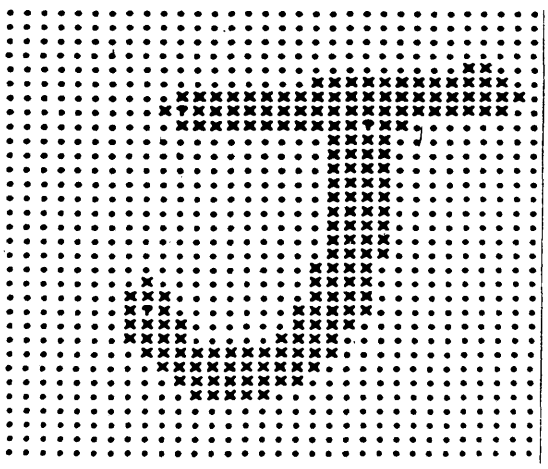


Fig 5.24 Experiment 10 : Tests Results : F4/F6 m/cs
Tasks : THIN , THICKEN

given set, this can also effectively reduce the number of cells in the memory matrix. Many features (and hence cells) are equivalent and consequently redundant. Only a relatively small proportion of the cells - corresponding to a 'preferred subset' of all possible features - need be retained. To illustrate this subset of features, consider again the 5-bit window as used above.

In this format, there is a maximum total of 32 unique features, all shown in Figs 5.20 and 5.21. However, if any feature that becomes redundant under rotation or reflection is removed from this list, the number of different cells drops to 12, as shown in Fig 5.25. These 12 features constitute the subset of 'preferred locations' in the memory matrix of this machine, and could in principle perform by themselves any isotropic picture processing task as competently as the full set of 32 cells.

It should be noted that these preferred features are in pairs. That is, each adjacent pair of features has the same neighbours, differing only in the centre point P0. If the dependence on P0 can be removed without detrimental effect on the test performance, then the number of cells required is halved - leaving only 6 distinct windows from the original 32. (It should be mentioned that this argument applies to the original 'full set' of features also - independence from P0 halves any memory matrix size.)

9-bit Window Memory Matrix Listing

In the case of the 9-bit F4 machine, the use of rotation and reflection of the input window (producing the F5 machine) reduces the total number of cells in the memory matrix from 512 to 102 preferred locations. A listing of all

'Preferred' Location Subset		'Non-preferred' Locations
Address	Feature	Equivalent Features obtained by Rotation and Reflection
00	. . .	
01	. X .	
02	X . .	04 . X . . 08 . . X 16 . . . X
03	X X .	05 . X X . 09 . X X 17 . X . X
06	X X . .	12 . X . X 24 . . X X 18 X . . X
07	X X X .	13 . X X X 25 . X X X 19 X X . X
10	X . X .	20 . X . . X
11	X X X .	21 . X X . X
14	X X . X .	28 . X . X X 26 X . X X 22 X X . . X
15	X X X X .	29 . X X X X 27 X X X X 23 X X X . X
30	X X . X	
31	X X X X	

Fig 5.25 5-bit Window 'Preferred Features' and Equivalents

these cells and their corresponding features appears in Appendix 2a.

In Appendix 2a, these data are laid out in the following manner. Each of the 512 cells in the memory matrix is assigned a hexadecimal address in the range 000-1FF. These are listed with the corresponding 3x3 bit window feature shown beneath in accordance with the re-formatting arrangement already described in Section 4.2. The preferred subset of features are 'boxed' throughout the listing. Examination of the 'unboxed' features reveals that all these features can be generated by rotation or reflection of those boxed, and consequently the former are redundant in isotropic picture processing. Again, note how these boxed features occur in pairs throughout the listing. This is because the addresses are organised with the least significant bit corresponding to P0 - the centre point value. Hence, the pairs of boxed features are identical apart from this central point.

In the F5 machine devised earlier, all 512 cells were retained despite the rotation and reflection of the input window. This was because of the resultant minimal changes required in the software simulator in the training phase. The machine rotated and reflected each input feature in training to one, two, four or eight equivalent features (dependent on its degree of symmetry) and modified all these cells. The testing phase looked up a single cell in the normal manner. The maintenance of the complete memory matrix (512 two-bit words) is a justifiable extravagance in this case, in the light of the advantages of this approach.

This software compatibility between machines with and without rotation and reflection extends to the listing of the addresses and data contents of the memory matrix. A sample listing of the cell data contents of the F5 machine as trained in Experiment 8 to 'thin' is shown in Appendix 2b. The format follows that of the window feature listings in Appendix 2a, and consists of each address listed above the corresponding cell contents : one of '.', '?' or 'X'. The preferred subset of cells are again boxed to aid interpretation. This is expanded upon below.

5.8 Interpretation of LPP Machine Algorithms

It has been noted in Section 2.9 that the examination of the memory matrix constitutes an attempt to interpret the machine's method of processing. The memory matrix contains the information derived from the training set, although it is in a form more suitable for assimilation by machine than man. However, it does represent the algorithm generated by the machine for processing pictures. This leads to two important possibilities :

1 the examination of the memory matrix contents (albeit hampered by their volume) gives insight into picture processing algorithms that come directly from a source of such processes - some examples. This is done without intervention by man and a consequent bias on how such a task should 'best' be performed. The algorithm contained in the memory matrix should reflect the training exactly, in the sense that it was produced autonomously by machine. If these data can be interpreted correctly, they may well represent a

definitive, rather than arbitrary, algorithm. This is ultimately what research into picture processing by machine is seeking to achieve. This problem has thus been reduced to :

(a) the provision of suitable examples,

(b) the correct interpretation of the resultant internal state of the machine.

2 the possibility of examining all possible algorithms with such a machine is approaching. With a machine that does not use rotation or reflection of a binary 9-bit window there are 2^{512} ($\sim 10^{150}$) possible different algorithms. It is obviously impossible to test these by automatic generation of all these algorithms. However, on using rotation and reflection this number falls to 2^{102} ($\sim 10^{30}$). This is still an impossibly large number, yet by selective additional restraints it can be reduced further to a practicable, although still large, number. Such restraints could be generated by considering the task attempted. The process of 'thinning' for example, need only be restricted to regions of the pattern (and hence windows) with a crossing number of two. That is, a window straddling an edge of an object has two changes of polarity of the neighbours surrounding the centre point when examined as a ring. The 20 pairs of features in a 9-bit window that fulfil this condition are indicated with an asterisk in Appendix 2a. If the dependence on the centre point P0 can again be removed (which again halves the number of features involved) the number of possible different features drops to 20 single examples. These are shown in Fig 5.26 below with P0 thus removed.

```
. . .     X . .     X . .     * X .
X   .     .   .     X   .     X   .
. . .     . . .     . . .     . . .

X X X     * X X     * X *     X X *
.   .     X   .     X   X     .   X
. . .     . . .     . . .     . . X

* X *     * X *
X   X     X   X     (P0 has been removed)
. . X     X . X         * - 'Don't care'
```

Fig 5.26 20 9-bit Windows with Crossing Number of Two

This leaves only 2^{20} ($\sim 10^6$) possible different algorithms, hence it would be feasible to attempt all these cases by automatic generation and testing of these memory matrices. Other such restraints on the features to be examined could be tried for other picture processing tasks. This would similarly enable the testing of an exhaustive set of memory matrices as a step towards interpretation of such machine generated algorithms.

5.9 Experiment 11 : The Variation of TP with Training Set Size

The Trained Percentage value (TP) has been used in Experiments 7 and 8 to determine how the changes in operation and format of the LPP machine alter the effective training received. Here the effect of a change in the

training set size on TP will be examined, and how this relates to the change in output when the machine is tested on actual patterns.

Experiment

An F5 machine was set up, with a tri-state memory matrix and a 9-bit window rotated and reflected in training. It was trained on a varying number of hand-drawn characters, taken from the set used earlier in Experiment 2. (These were digitised in a 32^2 format here, and not 16^2 as used earlier.) The example pictures were manually cleaned and thinned to a uniform limb width as described earlier. Five training runs were made, with training sets of 1, 4, 16, 32 and 100 pairs of patterns. After each training session, the memory matrix was tested with a fixed test IPP, thus generating a set of OPPs.

Results

The data collected from the five memory matrices are shown in Fig 5.27 as a table and histogram in the manner of Fig 5.9. A graph of TP against 'Ntr' (the number of pairs of patterns in the training set) is also included. The test IPP and the five OPPs produced are shown in Fig 5.28.

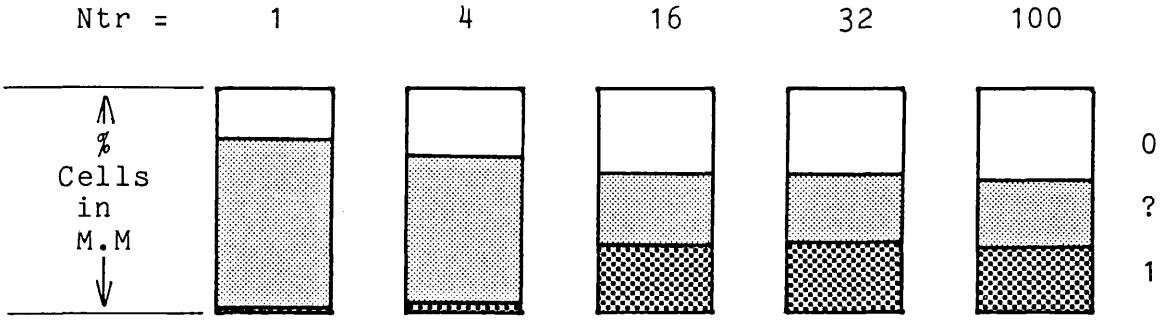
The TP shows a rapid rise as Ntr changes from 1 to 16, but levels off as Ntr exceeds 16. This is reflected in the OPPs - the 'quality' of processing increases rapidly as Ntr reaches 16, but thereafter remains relatively constant.

Discussion

The set of OPPs shows machine behaviour that becomes self-explanatory in the light of the TP data collected. The increase in TP reflects an increase in performance seen in

Ntr	% Cells = 0	% Cells = ?	% Cells = 1	T.P.
1	22.3	76.7	1.0	23.3
4	30.0	64.5	5.5	35.5
16	37.9	31.6	30.5	68.4
32	37.9	30.1	32.0	69.9
100	41.0	29.3	29.7	70.7

Histogram of Above Data : (Key as in Fig 5.9)



Graph of T.P against Ntr :

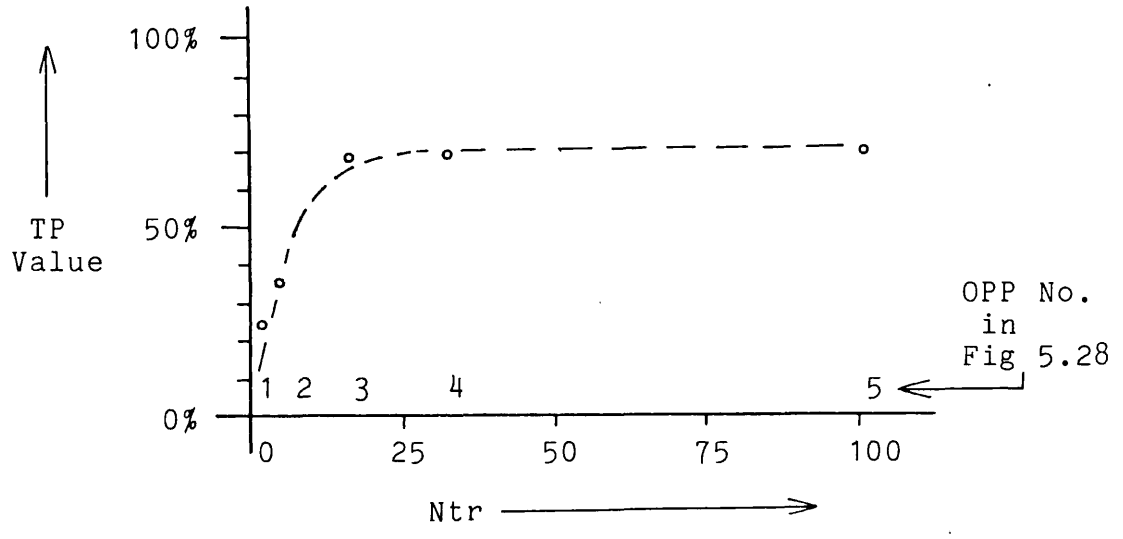
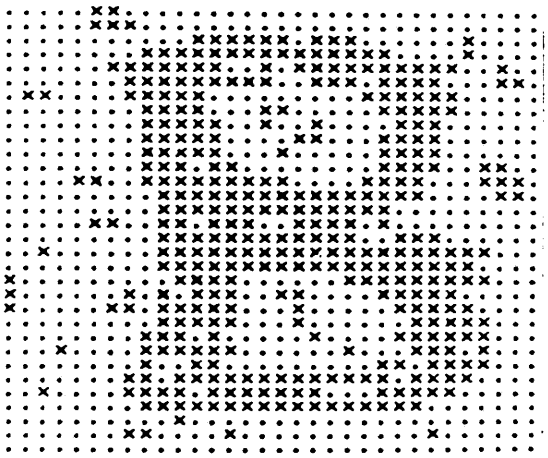


Fig 5.27 Distribution of M.M Contents of F5 Machine Trained in Experiment 11 with Different Ntr

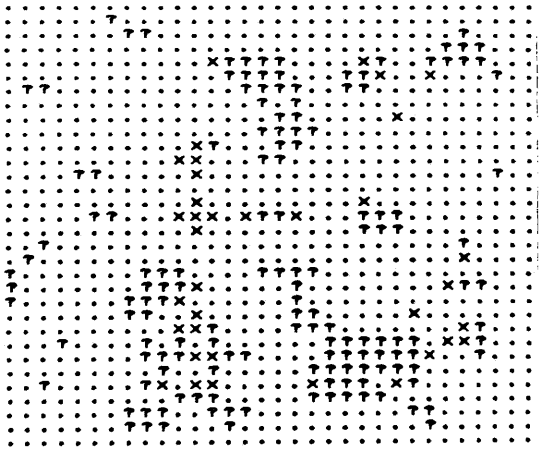
Test IPP

IPP All Test Runs

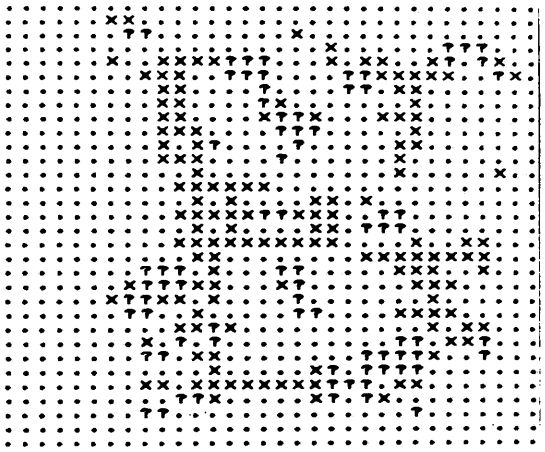


Test OPP Set

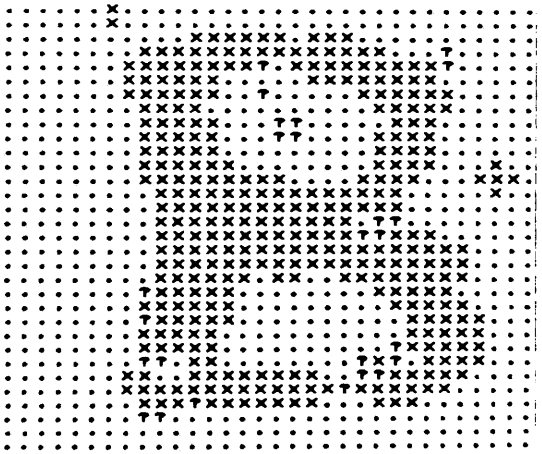
OPP 1 Ntr = 1



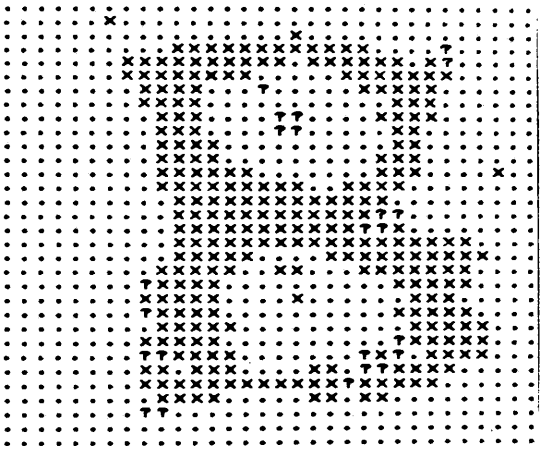
OPP 2 Ntr = 4



OPP 3 Ntr = 16



OPP 4 Ntr = 32



OPP 5 Ntr = 100

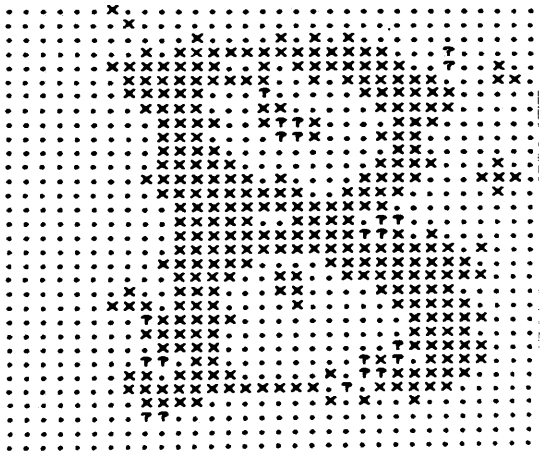


Fig 5.28 Experiment 11 : F5 Machine
Trained on 1, 4, 16, 32, 100 Pairs

the first three cases. Similarly, a relatively constant TP appears as a constant processing performance in the last three cases.

There only remains a need to explain the 'knee' in the curve in the graph of Fig 5.27. This can be attributed to the fact that of the 102 possible different features under rotation and reflection, there is a large variation in the likelihood of a particular feature occurring within a particular processing application. For example, those features having a large crossing number represent small width, multi-limbed junctions. These are obviously uncommon in this character set, and hence no amount of training is ever likely to set those memory matrix cells corresponding to such features. Since the TP curve levelled off around 70%, it may be inferred that 30% of the possible features are 'uncommon' in this character set. The memory matrix contents (after training with 100 characters) are listed in Appendix 2c. Examination of this listing reveals that those cells still in the initialisation state '?' after training with 100 characters (boxed in this listing) do represent uncommon features in the training set, as expected.

5.10 Experiment 12 : Down-Loaded Memory Matrix

It has been mentioned in Section 5.8 that an automatic generation of memory matrix contents could be used to test all possible picture processing algorithms. An alternative to this is the production of a single set of cells, generated externally and down-loaded into the LPP machine. The machine may then be tested in the normal manner to

reveal how this matrix performs.

The 'Keyin' Facility of the Simulator

This acts by initially clearing the memory matrix of the machine, then prompting the operator with each of the 102 preferred cell features. (This use of the subset of cells greatly reduces the effort required, without detracting fron the essential features of this method.) The operator replies with the data cell contents (one of '0', '?' or '1') to be placed in that cell corresponding to the feature presented. This reply is inserted into that cell, and also into all the equivalent cells under rotation or reflection. In this manner, all 512 cells are filled with data as the machine proceeds through the preferred subset. The resultant memory matrix may then be stored permanently, before testing begins.

Experiment

The 'keyin' facility was used to create a memory matrix designed to locate the edges or outlines of objects. This was done by consideration of each of the features presented to the operator, then responding with the new pixel that would ultimately leave only the edge of objects. That is, if the feature would be centred on an edge point, the '1' reply was made; if not, '0' was specified. Appendix 2d shows the resultant memory matrix - the cells prompted (the preferred subset) are shown 'boxed' together with the operator's replies of cell contents. Examination of Appendix 2d and the listing of the corresponding features (Appendix 2a) will show how the operator responded to each feature, in an attempt to generate the 'edgeing' function. (This LPP

machine was consequently to bypass the usual training period.)

The machine was tested in parallel with three objects illustrated on the left of Fig 5.29 to ascertain its performance. The resultant OPPs are shown on the right of the diagram.

Results

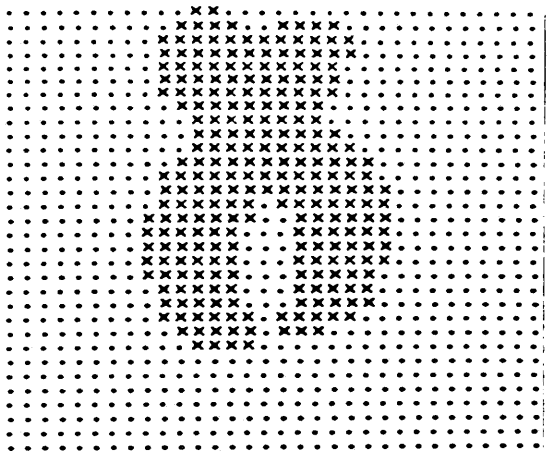
The 'keyed in' memory matrix can be seen to have correctly 'edged' the test IPPs - resulting in a single width line located on the outermost egde of the objects. (Interestingly, an 'error' has occurred at the bottom of OPP 3, shown boxed in Fig 5.29. This is generated from the input feature 'OBD' in IPP 3. Examination of this cell in Appendix 2d reveals that it does indeed contain an erroneous '0', fed in by the operator.)

Discussion

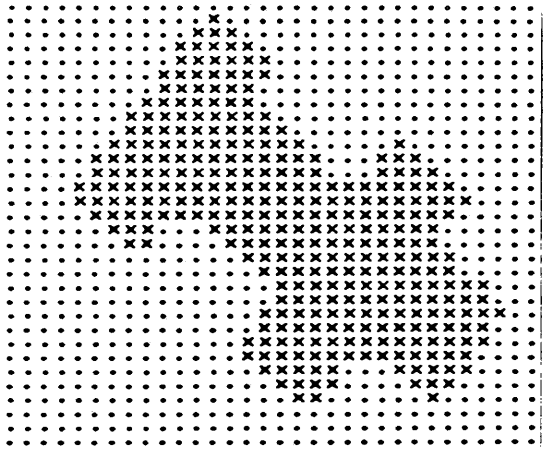
It has been shown that manually generating a memory matrix for a specific task is a practical alternative to training by example. The edging operator is chosen as particularly simple to implement, but serves as a suitable illustration of the principle. However, it requires the intelligence of the operator to supply the ability to process pictures. This procedure does not constitute machine learning in any useful sense, as the machine is being used simply as a tool for executing the process defined by the operator. Consequently, this avenue of research will not be developed further here. It will merely be recorded as an alternative mode of operation in which no machine learning or trainability is exploited.

Test IPPs

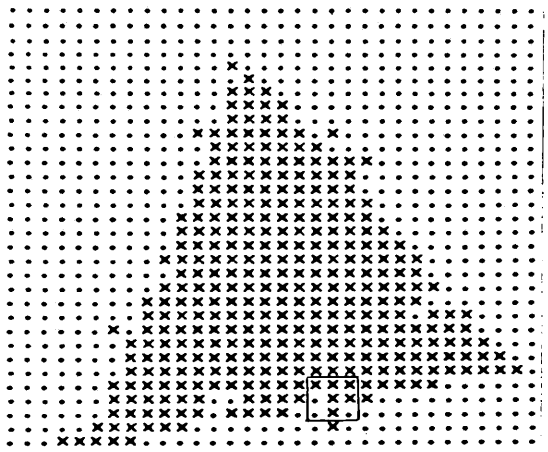
IPP 1



IPP 2

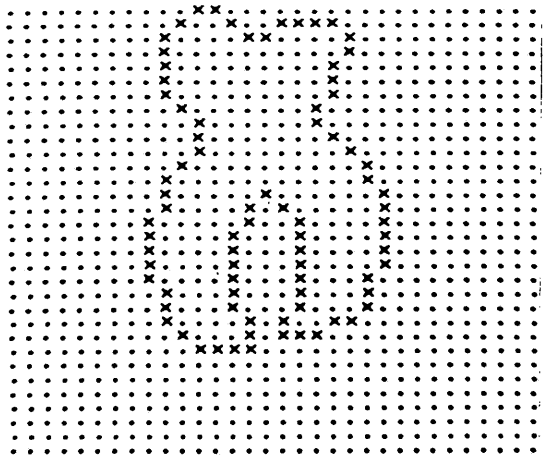


IPP 3

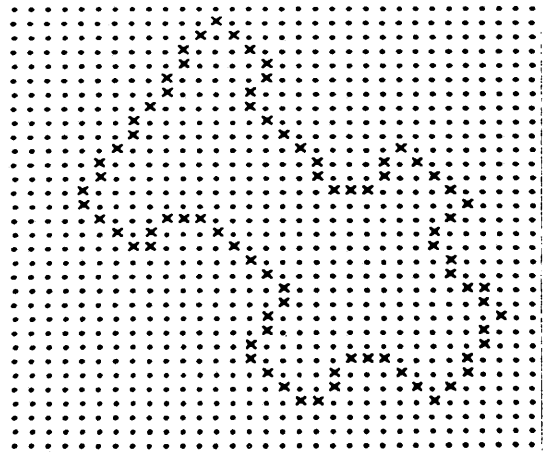


Test OPPs

OPP 1



OPP 2



OPP 3

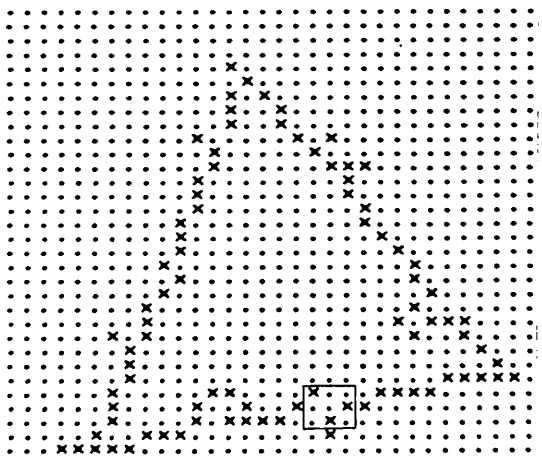


Fig 5.29 Experiment 12 : Memory Matrix
Keyed In by Operator to EDGE FIND

5.11 Summary of Experiments 6 - 12

To consolidate the results achieved in this second experimental chapter, there follows a brief summary of the main conclusions. It should be noted that all the algorithms described in this chapter have employed parallel processing in the testing phase.

A higher resolution (32^2) LPP machine was used in Experiment 6, with an otherwise identical internal format. This confirmed that such a machine can perform a set of different tasks by re-training alone.

Tri-state memory matrix cells were used in Experiment 7, enabling the machine to produce results which illustrate not only the algorithm, but also the adequacy of the training received. A 'trained percentage' (TP) was defined which served as a measure of training quantity.

Rotation and reflection of the input window in training resulted in an effective increase in training and thus in performance (Experiment 8). However, this is only of use in symmetrical picture processing. In Experiment 9 an assymetrical operation was attempted with such an arrangement - and illustrated behaviour heavily dependent on factors such as the scanning window direction. That is, the LPP is shown to retain recent training. If the training varies the machine is being 'asked to do the impossible' and hence becomes 'neurotic'. This will be examined later in Experiments 16 and 17.

In Experiment 10 a 5-bit window was used to allow the generation, and hence examination of a small (32 cell) memory matrix. This direct examination gives considerable insight into how the memory matrix reflects the training,

the use of the TP value, and the effect of rotation and reflection of the input window. This leads to the possibility of limited interpretation of the full sized (512 cell) memory matrix generated with a 9-bit window.

The effect of varying the training set size is examined in Experiment 11, where the OPP results and TP values were viewed in the light of this change.

Experiment 12 illustrated the possibility of down-loading the memory matrix, rather than training by example. This alternative produces good results, yet relies on a prior knowledge of the operation and layout of the machine.

Further experimental investigations will be made in the following chapter.

CHAPTER 6

EXPERIMENTS EMBODYING SPECIAL TRAINING TECHNIQUES

6.1 Experiment 13 : Training by Specially Prepared Examples

Two possible methods of increasing the quantity of training received by the machine were suggested in Section 3.5. The first was the increasing of the size of the training set, as demonstrated in Experiment 11 above. The second was the creation of special training characters to increase the number of different features found in each training pair. This latter course will be attempted here.

This problem can be reduced to providing special examples which illustrate the task to be performed more comprehensively than random example characters. These characters will ideally contain a larger proportion of all possible features, together with correct examples of the task as applied to these features.

Examination of the memory matrix after training on randomly chosen example characters will show the features for which training could be improved. These features not trained correctly invariably cause the generation of '?' or erroneous pixels in testing. So, a training example pair containing these features - shown processed correctly - should result in enhanced test performance. This will be attempted in the following experiment.

This experiment will also be used as an opportunity to attempt several other developments in machine layout and

operation. The high quality of training, as a result of using these specially prepared characters, allows changes to the machine in other areas without poor training masking the results.

6.1.1 The 'Thinning' Task

The task of thinning was chosen as a suitably non-trivial task, revealing a great difference between machines trained on random and specially created examples. This task was attempted by an F5 machine trained on random examples in Experiment 8. This same machine format will be used here. The training examples were shown at the top of Fig 5.3, and the test results in Fig 5.14. The most serious shortcoming was the breaking of characters' limbs, which occurred when an already thin limb was processed. This was to be expected, since such thin limbs were not present in the training IPP - and hence the machine was not trained to process such features. The essence of the approach to be attempted here is to create a training set containing examples of all possible features shown correctly processed, such that the machine can learn to deal with any test IPP presented to it.

The Creation of Special Characters as Thinning Examples

The characters to be created should contain features shown processed as below, to train the machine to 'thin' correctly :

(a) Wide limbs (that is, edges that extend beyond the range of a single 3x3 window) should be stripped back - both convex and concave edges,

(b) Narrow limbs should be thinned down only as far as single width limbs,

(c) Single width limbs (including single width junctions) must be left unbroken.
(It is this last condition upon which earlier attempts at thinning have failed.)

Later developments within this experiment will show a further condition to be of interest :

(d) Ends of lines should be maintained.

The training pair created contains examples of all the above conditions (a) to (c), and a second pair includes the (d) condition also. These will illustrate how, with the proper training, it is possible to be quite specific regarding the exact picture processing task required of the LPP machine.

Initially, only the former case, with the three requirements (a) to (c) will be examined. The equivalent picture processing task may be summarized as :

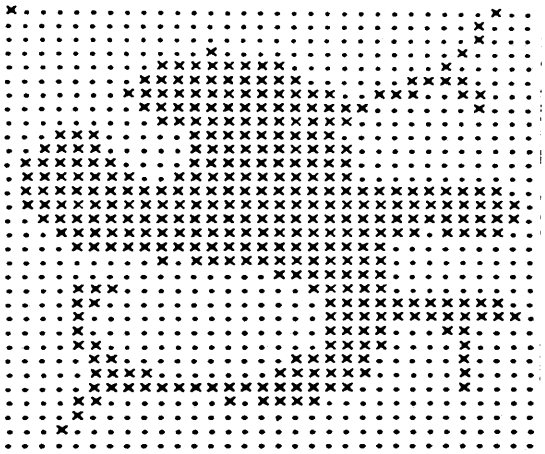
'Thin, whilst maintaining connectedness,
but losing line ends.'

The example characters were created by hand to illustrate this task, using the picture editing facility of the LPP simulator (see Chapter 7), and are shown in Fig 6.1a. These characters do not represent any particular pattern type, nor are they necessarily representative of the expected test set. They were designed solely according to the criteria (a) to (c) above. That is, IPP 1 contains thick

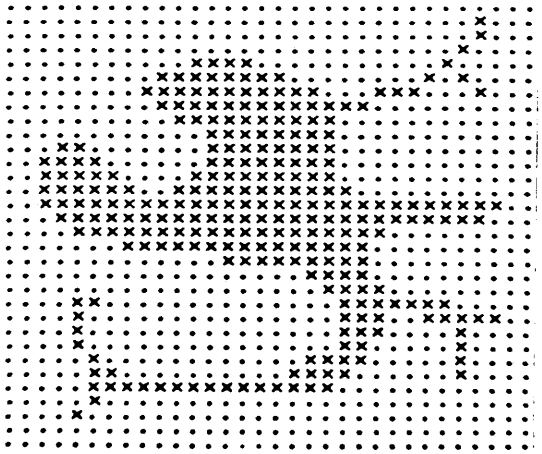
(a) Training Pair Specially Created to :

'Thin, Maintain Connectedness, but LOSE Line Ends'

IPP 1 Train



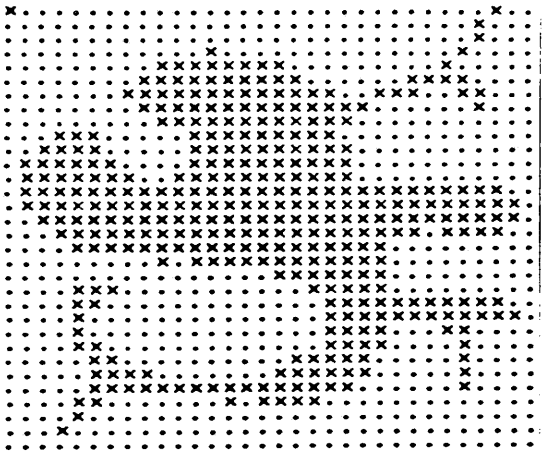
EXP 1 Train



(b) Training Pair Specially Created to :

'Thin, Maintain Connectedness, and KEEP Line Ends'

IPP 1 Train



EXP 2 Train

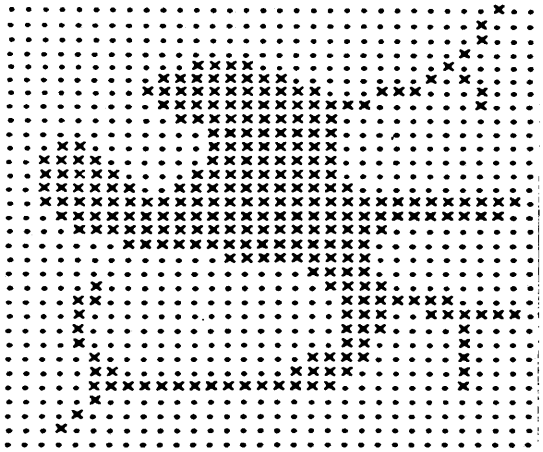


Fig 6.1 Training Characters Specially Created
For Experiments 13 (a-f)

limbs, concave and convex edges, narrow limbs, line ends, single width limbs and junctions, all shown correctly processed in EXP 1.

6.1.2 Experiment 13a : Initial Run

An F5 machine was trained to 'thin, maintain connectedness and lose line ends' using the pair of patterns shown in Fig 6.1a. The machine was then tested in the parallel mode with a set of three characters shown on the left of Fig 6.2. The resultant OPPs are shown on the right of this figure.

Results

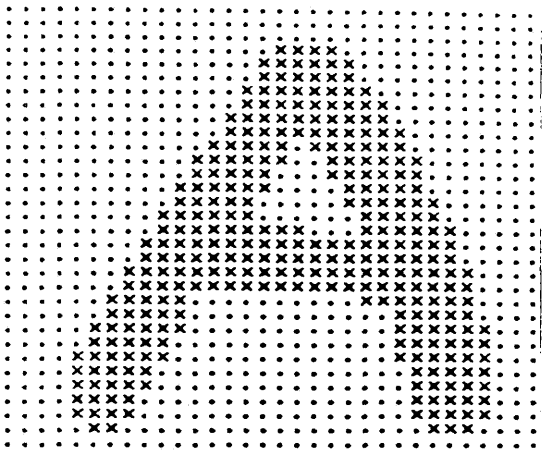
The machine can be seen to have thinned these three characters, although the last two have been broken. This has occurred in several places, where the original limbs were two units wide. This is apparently in conflict with the training stimuli, which did not show any examples of such behaviour.

Discussion

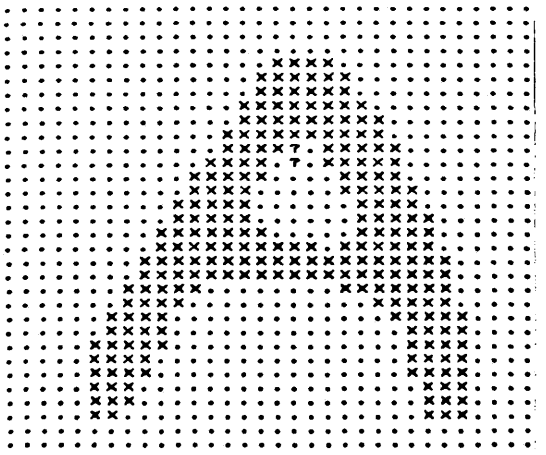
The reason for the breaking of the characters lies in the fact that these breaks only occur on double width limbs, the wider and narrower width limbs all remaining intact. This suggests that the breaks are due to the machine's thinning of these limbs simultaneously from both sides (as this particular testing was implemented in parallel), resulting in a zero width (broken) section. The machine correctly recognized a single width limb as a feature not to be further thinned from either side. The problem lies in the fact that the machine is effectively operating simul-

Test Inputs F5 Machine Test Outputs

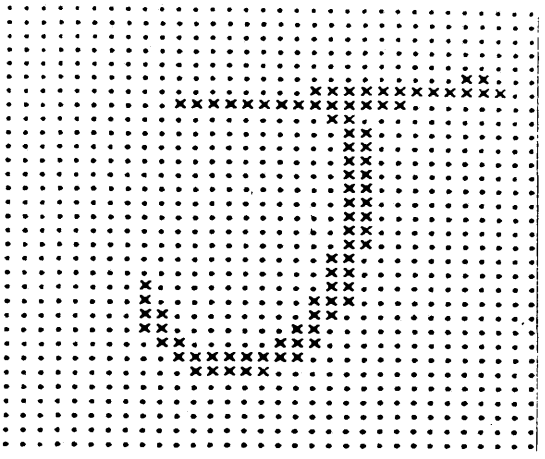
IPP 1



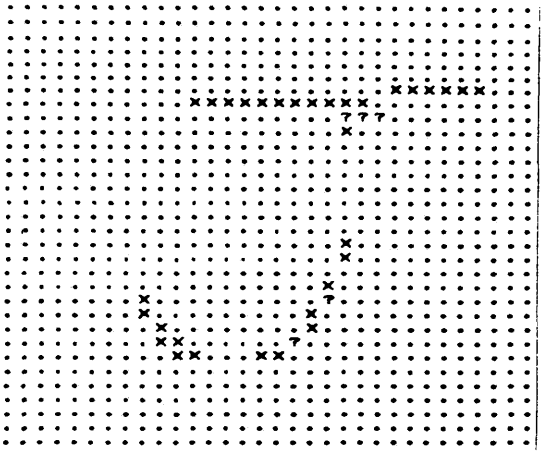
OPP 1p



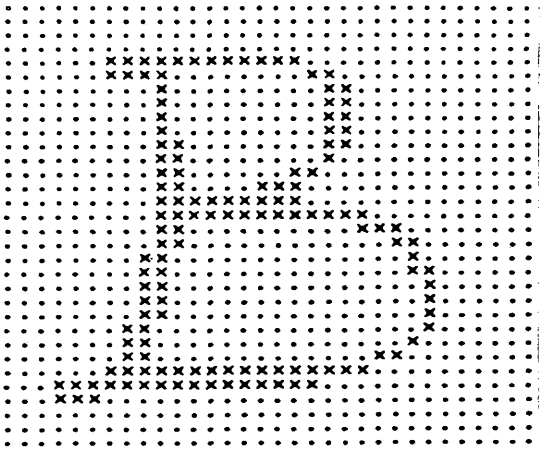
IPP 2



OPP 2p



IPP 3



OPP 3p

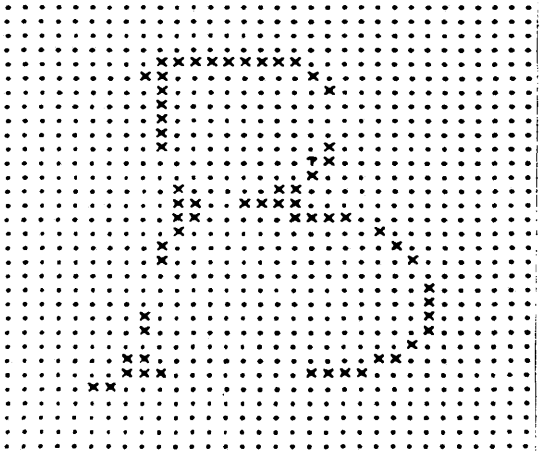


Fig 6.2 Exp 13a : Test Results after Training With Special Characters (One Pass, Parallel Mode)

taneously on all pixels in IPP, not that the training is necessarily incorrect. This breaking of a double width limb is a well-known problem when a picture processing machine is applied in this 'parallel' mode, as here (73,6). The alternative 'sequential' mode of operation was described in Section 4.3, and is attempted experimentally below.

6.1.3 Experiment 13b : Sequential Mode of Operation

The above experiment was repeated with all parameters and data kept constant, except that in testing, the LPP machine was applied sequentially to the test IPPs, to generate a new set of OPPs in Fig 6.3.

Results

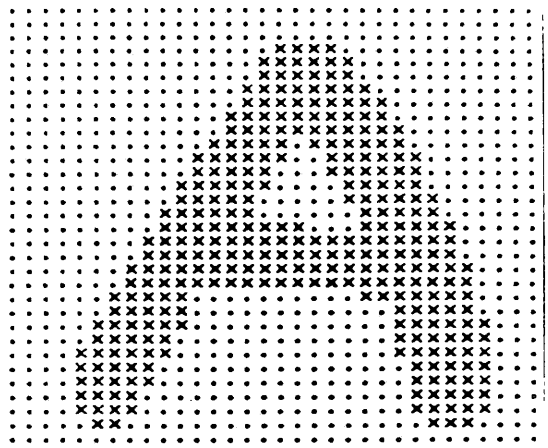
Here, the machine has again thinned the characters, including double and single width limbs, to a minimum of one unit width resultant limbs. However, in the case of the letter 'J' the upper left limb has been removed entirely, as has the upper left serif of the letter 'B'. The thick limbs of the letter 'A' have been thinned to a varying degree, leaving vertical left hand edges where areas of the object are 'shadowed' from the downward moving scan direction.

Discussion

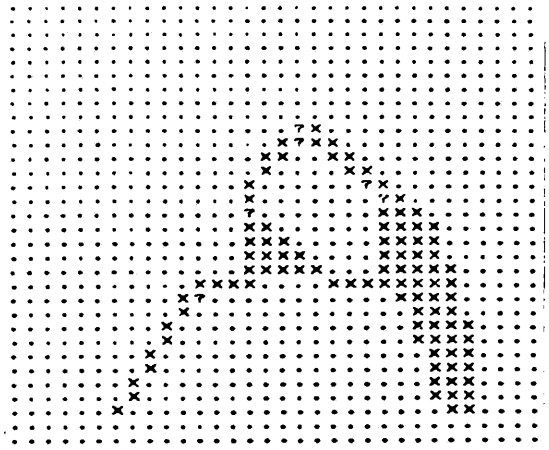
The sequential mode of operation has disabled the machine from breaking limbs. There are no breaks anywhere in the OPP set. However, the removal of entire limbs that point into the direction of scan is the apparent result of this sequential application. As the scan proceeds along a limb, it repeatedly removes points, as it has been trained to 'lose line ends'. This proceeds until either a junction is

Test Inputs F5 Machine Test Outputs

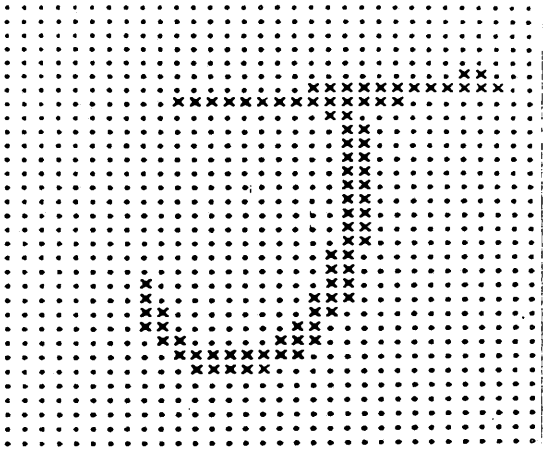
IPP 1



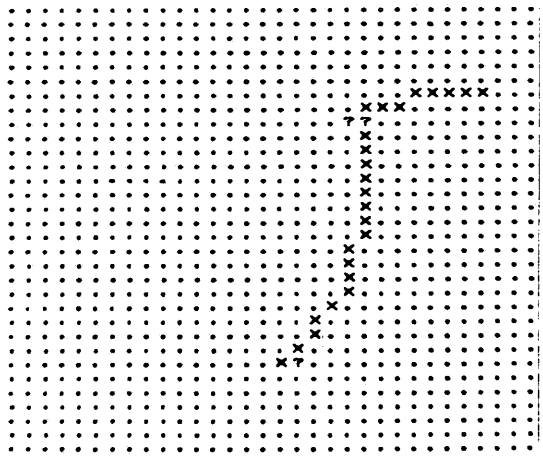
OPP 1s



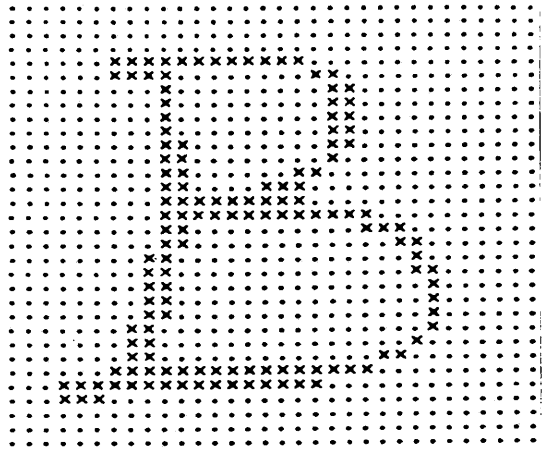
IPP 2



OPP 2s



IPP 3



OPP 3s

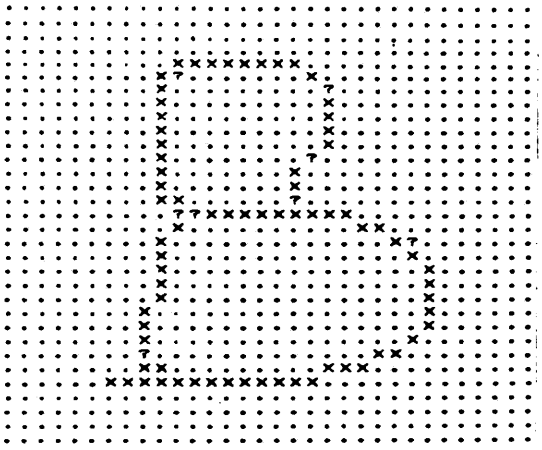


Fig 6.3 Exp 13b : Sequential Application of Machine Trained on Special Characters (One Pass)

encountered, or the scanning centre leaves the limb behind. It should be noted that while the limbs pointing into the direction of scan are removed entirely, the other limbs are only shortened by a small amount, as the machine 'loses a line end' on one occasion - when the scan is centered on this line end point.

This process of losing line ends, either by degrees or completely in one pass, indicates that the machine would eventually entirely remove any limbs with ends if allowed to repeatedly act on such patterns. This will be verified below.

6.1.4 Experiment 13c : The Use of Feedback in Testing

The F5 machine as trained above, is applied to the IPP set repeatedly until the OPPs show no further change. That is, feedback passes in testing will be made, as described in Section 4.4. The sequential mode of operation will be used again. The same IPPs are used, and the OPPs generated after each pass are illustrated in Figs 6.4 (letter 'A') and 6.5 (letters 'J,B').

Results

The letter 'A' is rapidly thinned to a unit width 'skeleton' after 4 passes through the machine. Thereafter, each pass removes a unit length from the lower limbs. The upper loop does not break at any time, although the junctions between this loop and the lower limbs contain '?' pixels. After 13 passes, the lower limbs have been removed entirely, leaving only the static single width upper loop.

The letter 'J' has its upper left hand limb removed

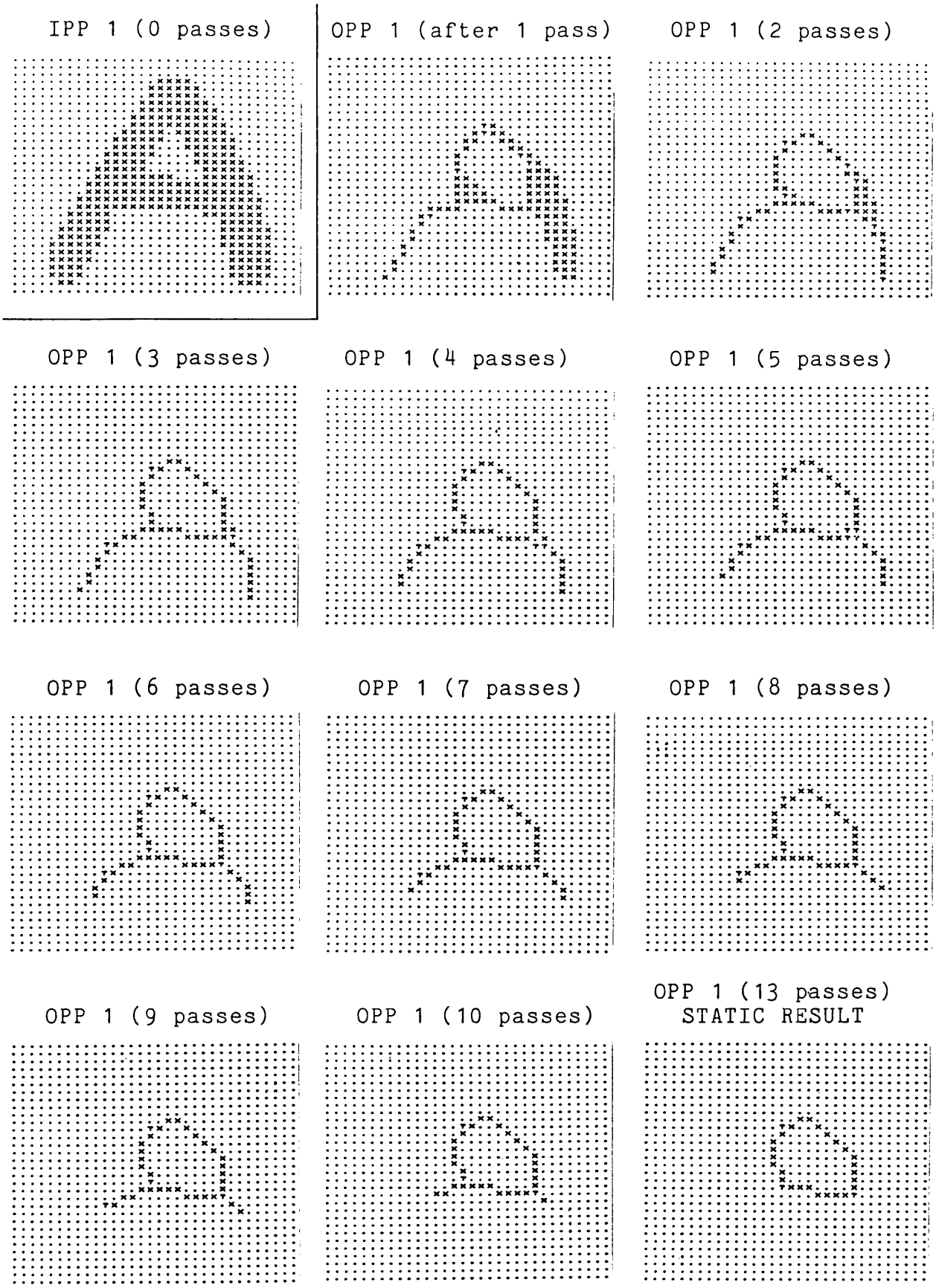


Fig 6.4 Exp 13c : Feedback Stages Applied
To IPP 1 in the Sequential Mode

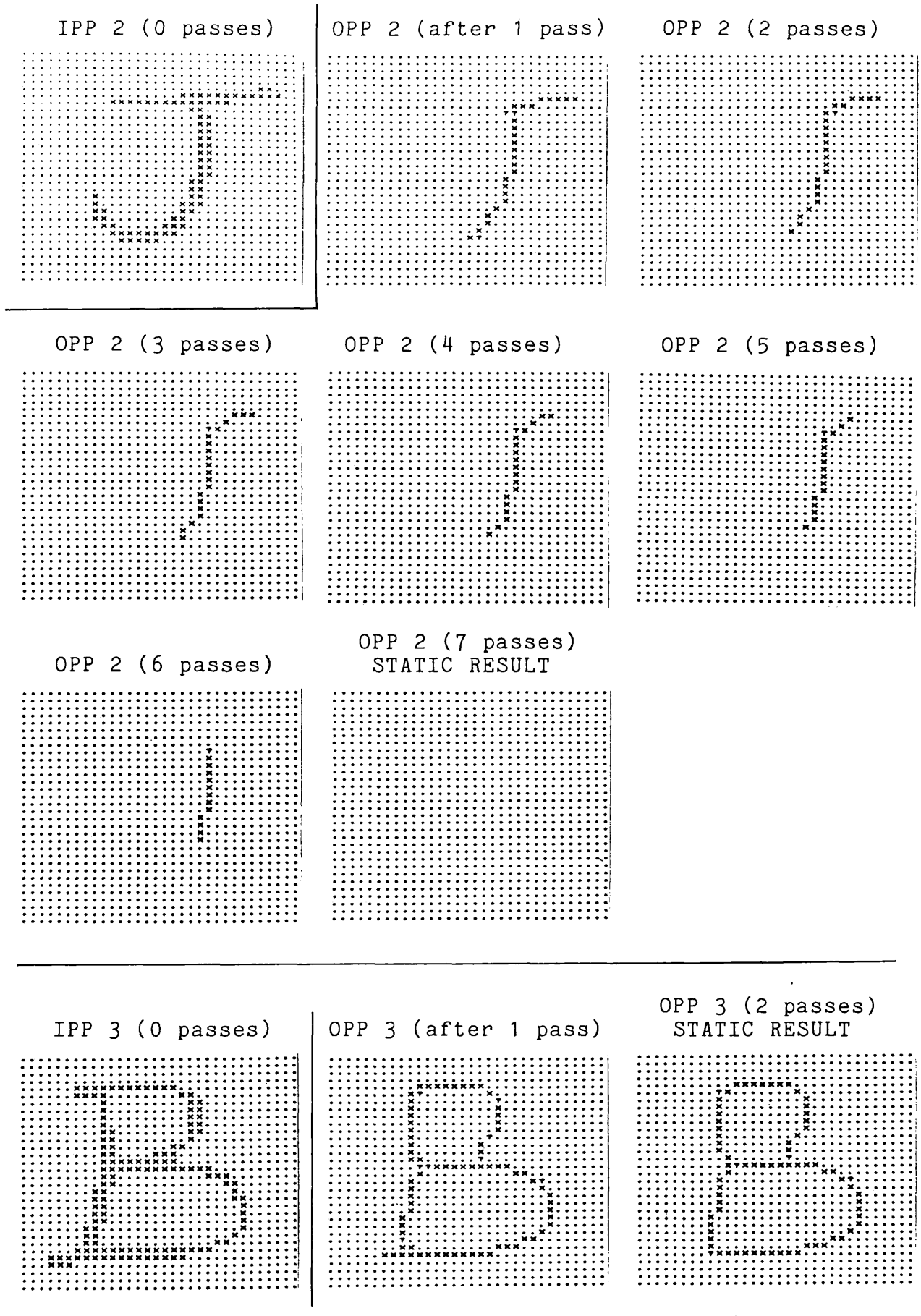


Fig 6.5 Exp 13c : Feedback Stages Applied
To IPPs 2 and 3 in the Sequential Mode

immediately after one pass. Thereafter the other limb ends are gradually cut back on each successive pass, until finally the direction of the upper right limb changes from horizontal to diagonal. This enables its complete removal in the next pass, and the consequent removal of the entire remaining vertical limb in the pass after that, leaving a null field.

The letter 'B' is quickly transformed into a single width skeleton, and after two passes such 'limbs' that do exist (the serifs) are removed. This leaves a static single width skeleton composed of two attached closed loops.

Discussion

As expected, by inference from the previous experiment, the repeated passing of patterns through this machine will eventually lose any features except closed loops. These will be reduced to single width loops, which are never broken. These loops are centered on the right hand lower edges of wide limbs due to the scanning direction effect, and hence are not ideal skeletons, which would be centered along the middle of such limbs. However, the process acts reliably and predictably, after training on just one specially produced pair of examples. The loss of line ends is to be expected as the training contained examples of this. The training will be modified in this respect below.

6.1.5 Experiment 13d : Modification to Training Set

In this section, condition (d) above is included - the retention of line ends - in order to produce a more conventional thinning algorithm. A pair of training patterns

was produced (Fig 6.1b) representing the algorithm :

'Thin, maintain connectedness, and KEEP line ends.'

On re-training the F5 machine, the same test set was presented to the machine. Again, the results were fed back repeatedly through the machine operating in the sequential mode, until the OPP results were static. The IPP letter 'A' and its set of OPPs at each pass are shown in Fig 6.6, the letters 'J,B' and their OPPs in Fig 6.7.

Results

A comparison of these OPPs with those produced in the previous experiment (Figs 6.4 and 6.5) show two main differences :

1 The retention of limbs at their full original length

2 The fewer number of passes required to reach a static result

In all three cases in this experiment a static, single width skeleton, with all limbs retained completely, was produced after a small number of passes through the machine.

Discussion

The modification to the training made in this experiment to retain line ends has manifested itself in the test phase as a retention of full length skeletons. This successful production of a unit width skeleton is well-known to be of great value in picture processing (51).

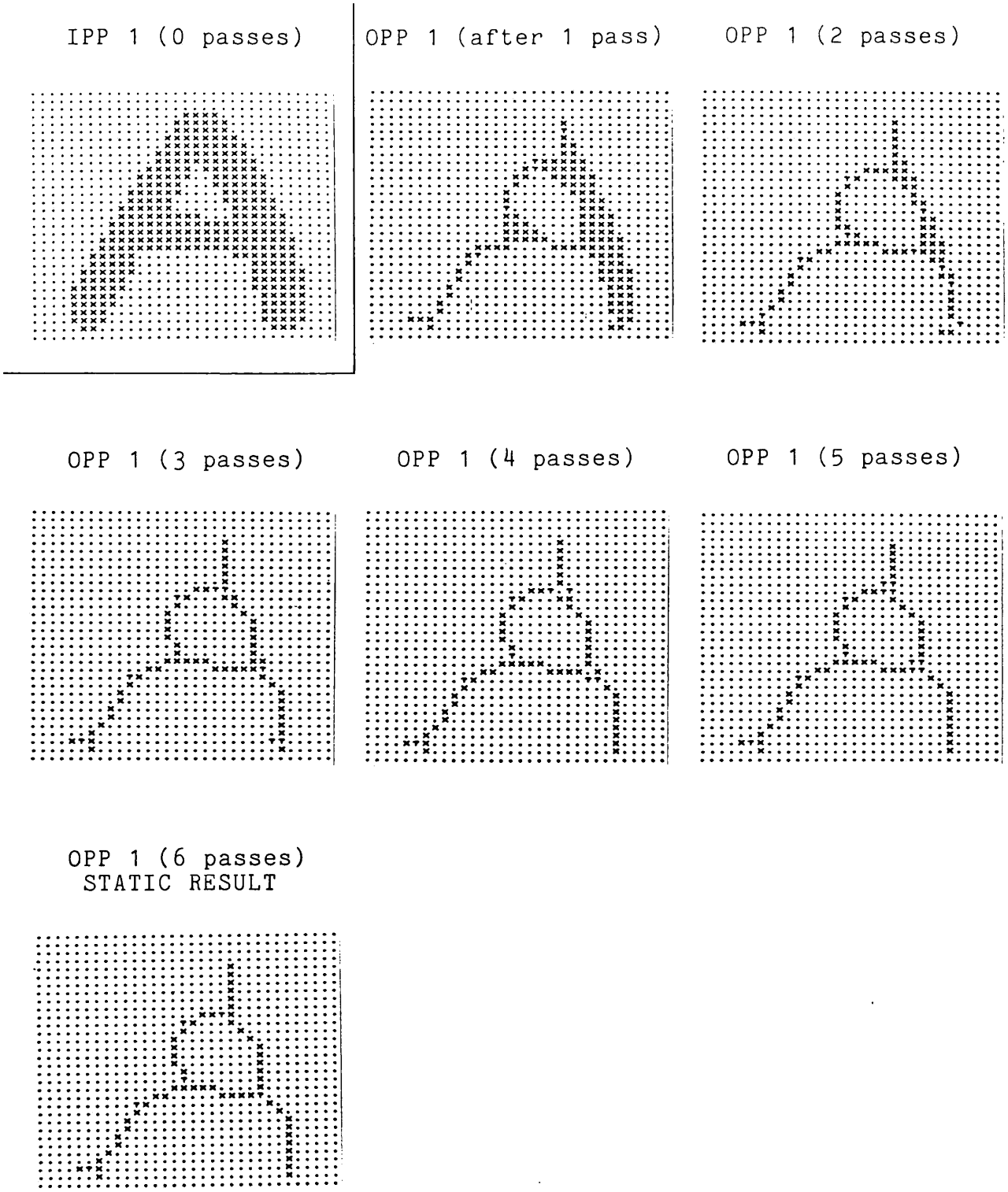


Fig 6.6 Exp 13d : Machine Trained To 'Keep Line Ends'
Feedback Passes Applied Sequentially to IPP 1

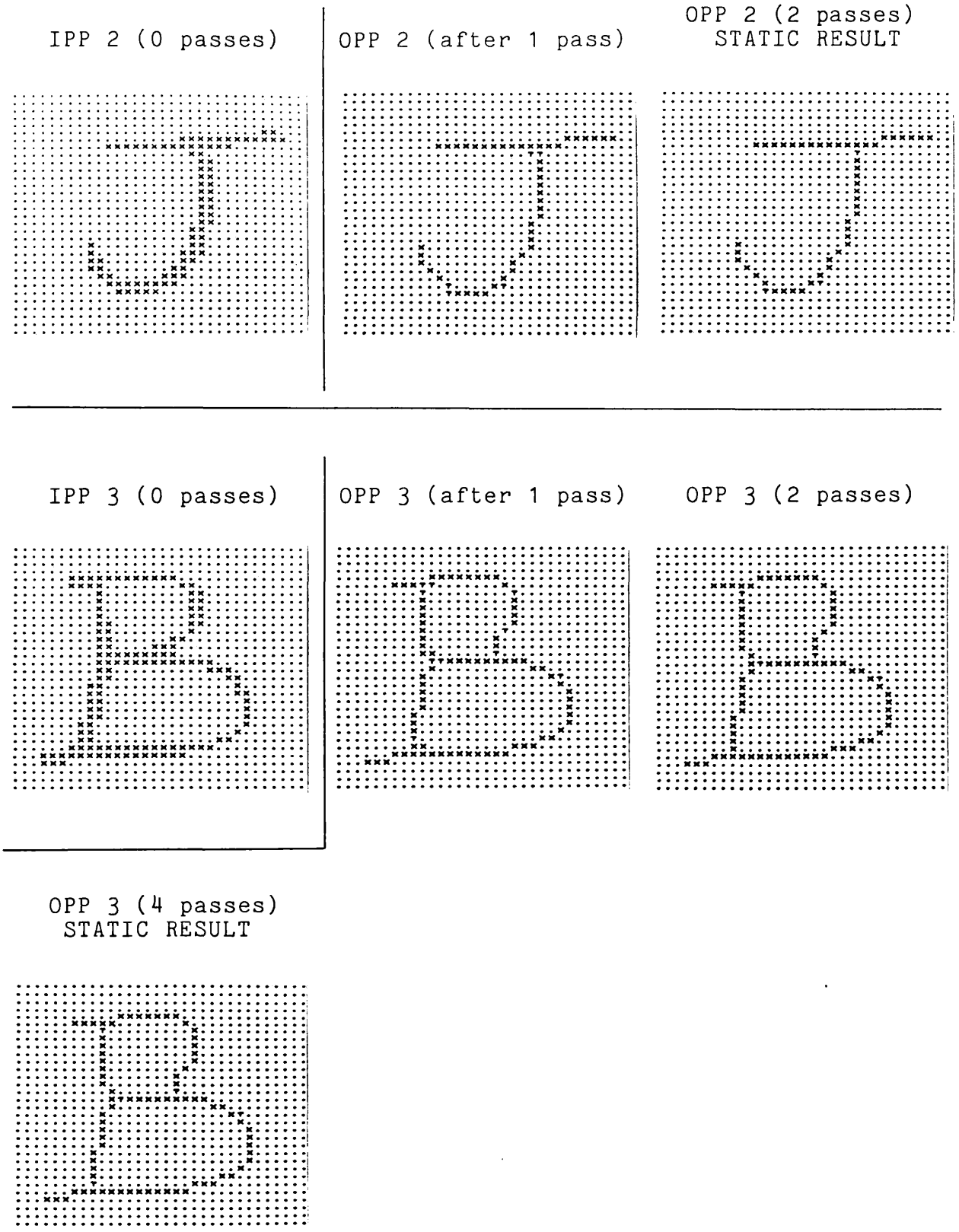


Fig 6.7 Exp 13d : Machine Trained to 'Keep Line Ends'
Feedback Passes Applied Sequentially To IPPs 2 and 3

6.1.6 The Examination of the Memory Matrices after Training with Special Purpose Examples

It has been illustrated above how it is possible to be quite specific - by training alone - regarding the exact picture processing task required of a LPP machine. In the cases above, the training has been adjusted to represent the two tasks :

1 'Thin, maintain connectedness, lose line ends' and
2 'Thin, maintain connectedness, keep line ends'

The two memory matrices after training with these tasks were examined as in the previous analysis; the resultant totals of cells set to '0', '?' and '1' being shown in Fig 6.8a as a table and histogram.

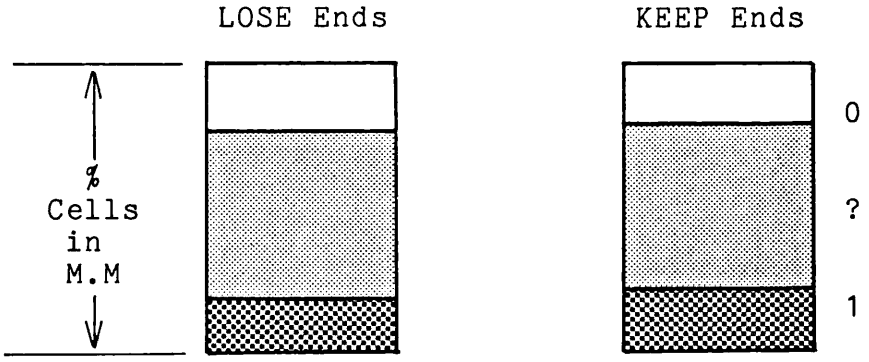
It should be noted that the TP value in both cases is 42.8% - a high value, compared to the TP value obtained in Experiment 8 (Fig 5.15) for the corresponding 'thin' task of 16.0%. This latter figure was obtained on training with a pair of non-specially prepared characters seen in Fig 5.3. This suggests that the training pairs used here (Fig 6.1) contain a much larger proportion of all possible 3x3 bit window features.

The fact that the two TP values for the two cases of different training here are exactly equal is to be expected. As seen before (Section 5.4), both pairs of training patterns used identical IPP patterns (on the left of Fig 6.1), it is only the EXPs that differ. IPP addresses the memory matrix, and if identical IPPs are used in different training phases they will still address exactly the same (and hence same number of) memory matrix cells in each case.

(a) Totals of M.M Cell Contents after Training :

TASK	No. of Cells=0	No. of Cells=?	No. of Cells=1	No. of Trained Cells
Thin, Maintain Connectedness, LOSE Line Ends	122 (23.8%)	293 (57.2%)	97 (19.0%)	219 (42.8%)
Thin, Maintain Connectedness, KEEP Line Ends	106 (20.7%)	293 (57.2%)	113 (22.1%)	219 (42.8%)

Histogram of Above Data : (Key as in Fig 5.9)



(b) Feature Differences in M.M. Cell Contents :

Preferred Features		. X . . X	. . X . X	. X X . X . . . .
No. of Equivalent Cells (Total=16)		4	4	8
Cell Data Contents	LOSE Ends	.	.	.
	KEEP Ends	X	X	X

Fig 6.8 Totals and Differences in Memory Matrices Trained in Two Cases in Experiment 13

As a result, the TP values will be identical. This can be seen in the totals of Fig 6.8a, where the number of cells left equal to '?' (untrained) is equal in both cases to 293 out of 512.

Differences in Memory Matrices Reflecting Different Tasks

The differences in the memory matrices after training on the two different tasks can be found by direct examination of the two cell contents' listing similar to those in Appendices 2b to 2d. The two complete memory matrices from this experiment are not reproduced here, but merely their differences. These differences occur in 16 cells, as may be deduced from the differences in totals in Fig 6.8a of cells equal to '0' or '1'. It should be recalled that the F5 machine was used, which rotates and reflects the input windows, and so these 16 cells correspond (in this case) to just three preferred cells or features. These features are shown in Fig 6.8b.

These differences are consistent with the training, in that the features where the training differs represent line ends. In the first case '0' pixels would be generated in testing (losing line ends) and, in the second '1' pixels are specified (keeping line ends).

This illustrates how the creation of special characters for training can generate quite specific memory matrices. In these two cases, the memory matrices differ in only 16 out of 512 cells (~3%), yet the resultant test OPPs generated show significant and predictable differences.

6.1.7 A Summary of Parallel/Sequential Processing Applied To Lose/Keep Line Ends

The experiments performed so far have given insight into the use of an LPP machine trained on two alternative sets shown in Fig 6.1 and operated in two alternative modes - parallel and sequential. Together with more exhaustive experiments not reproduced here, this has enabled some general conclusions to be drawn regarding these variations in operation and their resultant test performances.

The four cases of operation to be compared are :

1 Parallel operation - trained to lose line ends,
2 Parallel operation - trained to keep line ends,
3 Sequential operation - trained to lose line ends,
4 Sequential operation - trained to keep line ends.

In all cases the OPPs are repeatedly fed back to the inputs in testing to observe the processing develop to its ultimate result.

Experimental Investigations

Experiments 13c and 13d have respectively examined cases 3 and 4 above, and further experiments (not reproduced here) have examined cases 1 and 2. These results from cases 1 and 2 may also be deduced from the behaviour seen in Experiments 13a, and will be summarized below. Examples of cases 2 and 4 (processed to a static result) will be compared later in Experiment 13f.

Comparison

The summary of the behaviour of the four cases appears in Fig 6.9. This includes a description of the changes in test patterns as they pass repeatedly through the machine, and hand-drawn sketches of OPP sets illustrating the process.

From this diagram, the test performances in the four cases can be summarized as follows :

1 Parallel operation - trained to lose line ends

Breaks character, then loses resultant line ends, finishes with a null field from any starting object shape.

2 Parallel operation - trained to keep line ends

Breaks character, but retains resultant line ends, finishes with possible short breaks in an otherwise complete unit width skeleton.

3 Sequential operation - trained to lose line ends

Cannot break character, but loses line ends if any, finishes with only closed loops remaining as unit width rings.

4 Sequential operation - trained to keep line ends

Cannot break character or lose line ends, finishes with a complete unit width skeleton; often over-elaborate as 'limbs' are produced to corner points of original thick limbs or noise spurs.

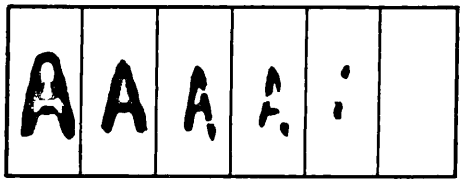
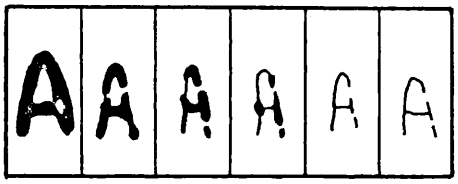
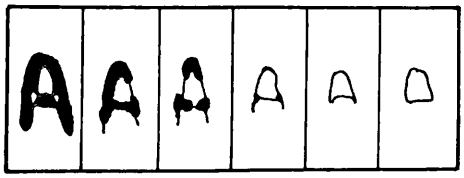
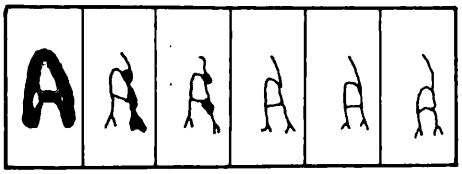
MODE	M/c Trained to : Thin, while LOSING Line Ends	M/c Trained to : Thin, while KEEPING Line Ends
PARALLEL	Parallel operation, hence double width lines CAN be broken as machine strips away two sides simultaneously. Most lines eventually ARE broken, and ends that result are then gradually trimmed back on successive passes. Result: NULL Field. In general, objects fragment, then line ends retreat: 	Parallel operation also, hence breaks CAN occur as at left. However, line ends that result on either side of such breaks are retained once they (and other) limbs reach unit width. Result: Skeleton correct but for short breaks Objects thin to a broken skeleton: 
SEQUENTIAL	Sequent. operation means NO breaks. No loops or limbs are broken, but once limbs are of unit thickness, they disappear (QUICKLY if pointing into the scan direction, SLOWLY if pointing away from the direction of scan.) Result: single width loops with no limbs. Characters without loops result in NULL field. Skeleton produced, limbs disappear : 	Sequent. operation means NO broken limbs possible Once unit width reached, (often after first pass) ends retained. Hence, over-elaborate skeleton produced with limbs to corners and noise spurs. Result: full-length complete skeleton centred on lower right edges of original. Objects quickly reach elaborate skeleton : 

Fig 6.9 Comparison of Machine Operations in Parallel or Sequential Mode, Trained to Lose or Keep Line Ends

6.1.8 Experiment 13e : Direction of Limbs Affecting Processing Time

It has been stated in Fig 6.9 that in the case of a sequential LPP machine trained to lose line ends (case 3 above), the direction of the limbs compared with that of the window scanning direction affects the time taken to remove these limbs. This will be examined here.

An F5 machine was trained on the pair of patterns in Fig 6.1a ('lose line ends') and then repeatedly applied sequentially to a test IPP. This test pattern was composed of two objects in the same field, essentially identical apart from their orientation. The objects each consisted of a thick closed loop with a thick limbed spur projecting from it - upwards in one case (into the scan direction) and downwards in the other (with the scan direction).

This IPP 4 is shown in the top left hand corner of Fig 6.10. The resultant OPPs after successive passes through the machine are shown in the remainder of the diagram.

Results

The thick limb pointing upwards is removed entirely after just one pass through the machine. The remaining two loops are reduced to unbroken single width static rings after three passes. The lower limb is gradually removed, at the rate of one pixel each pass, and finally disappears completely after nine passes, leaving only the two static loops.

Discussion

The immediate removal of the upward pointing limb is a result of a sequentially applied algorithm, continually

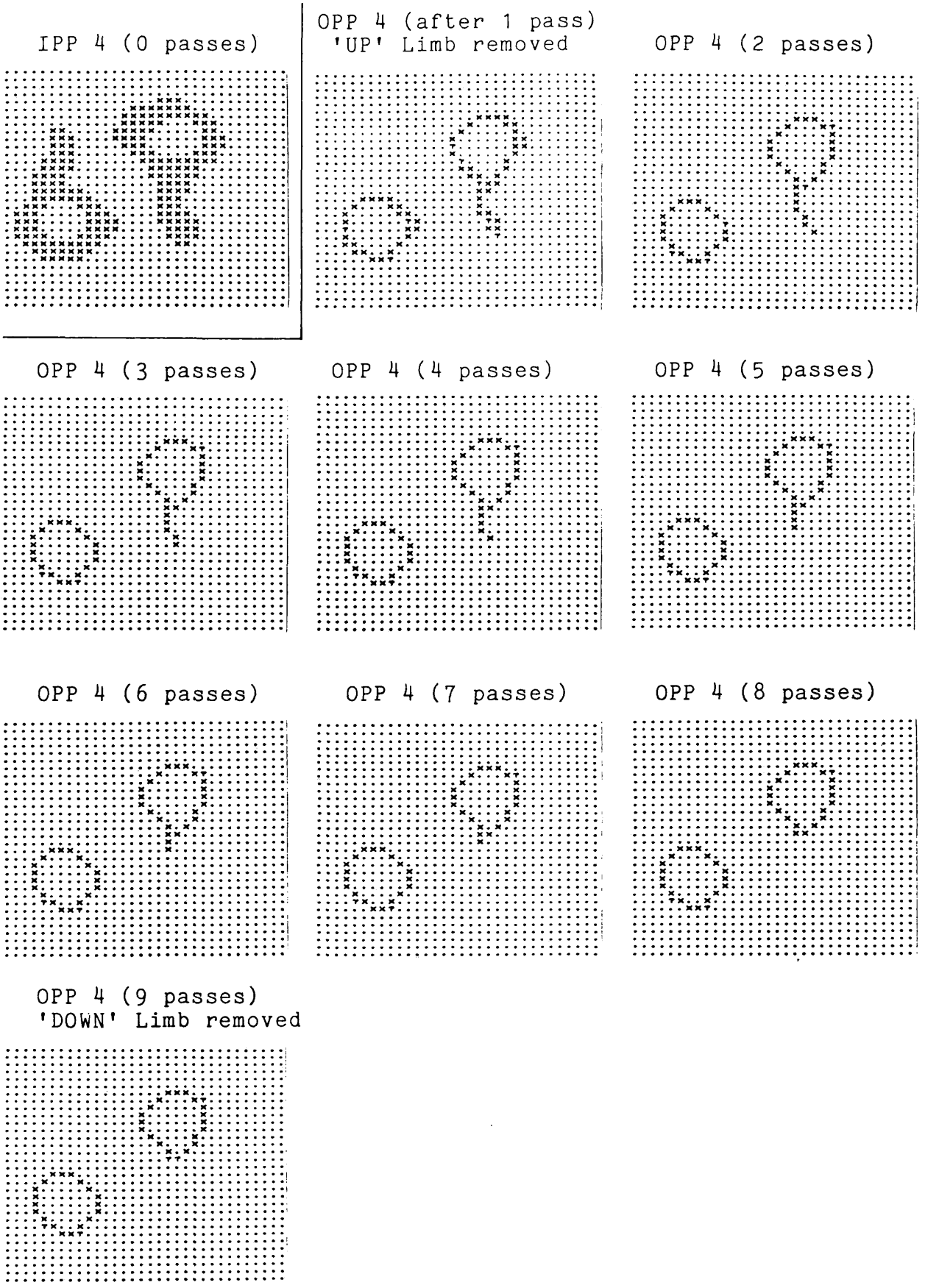


Fig 6.10 Exp 13e : Limb Direction Affecting Removal Speed

removing the line end, then on the next line scan (within the same field scan) removing the already retreating line end. This process repeats until the entire limb is removed in one frame scan - that is, in one pass through the machine. The downward pointing limb loses only a small end portion on each pass, as the scan only covers the line end once in each field scan or pass.

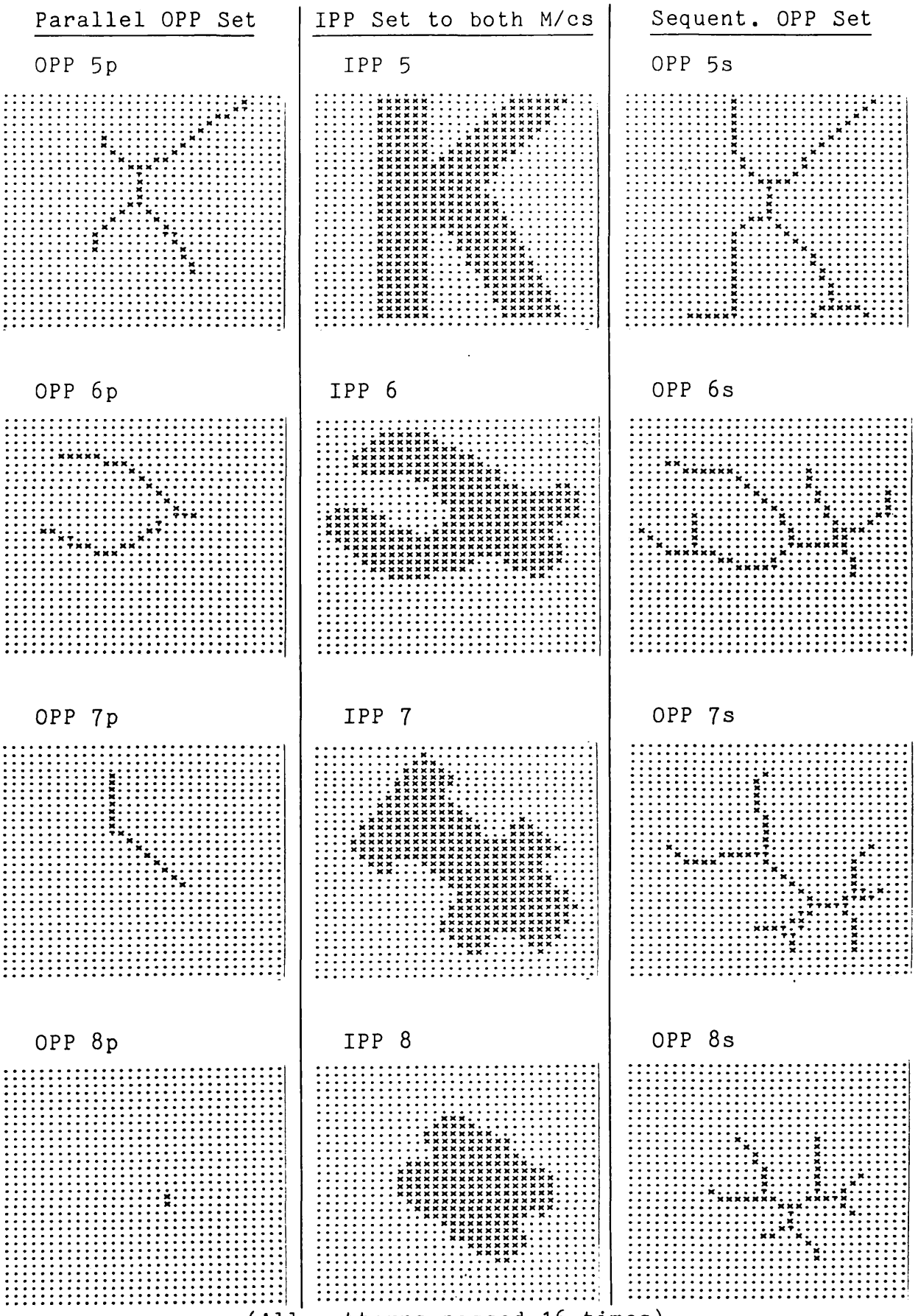
This result confirms the performance behaviour claims made in Fig 6.9 regarding the case 3 machine. This dependence on the scan direction suggests that a picture processor could take advantage of a reversible scan.

6.1.9 Experiment 13f : Parallel/Sequential Behaviour Exhibited With a LPP Machine

To confirm the claims made in Section 6.1.7 regarding the behaviour of sequential and parallel machines, this final section of this experiment was run.

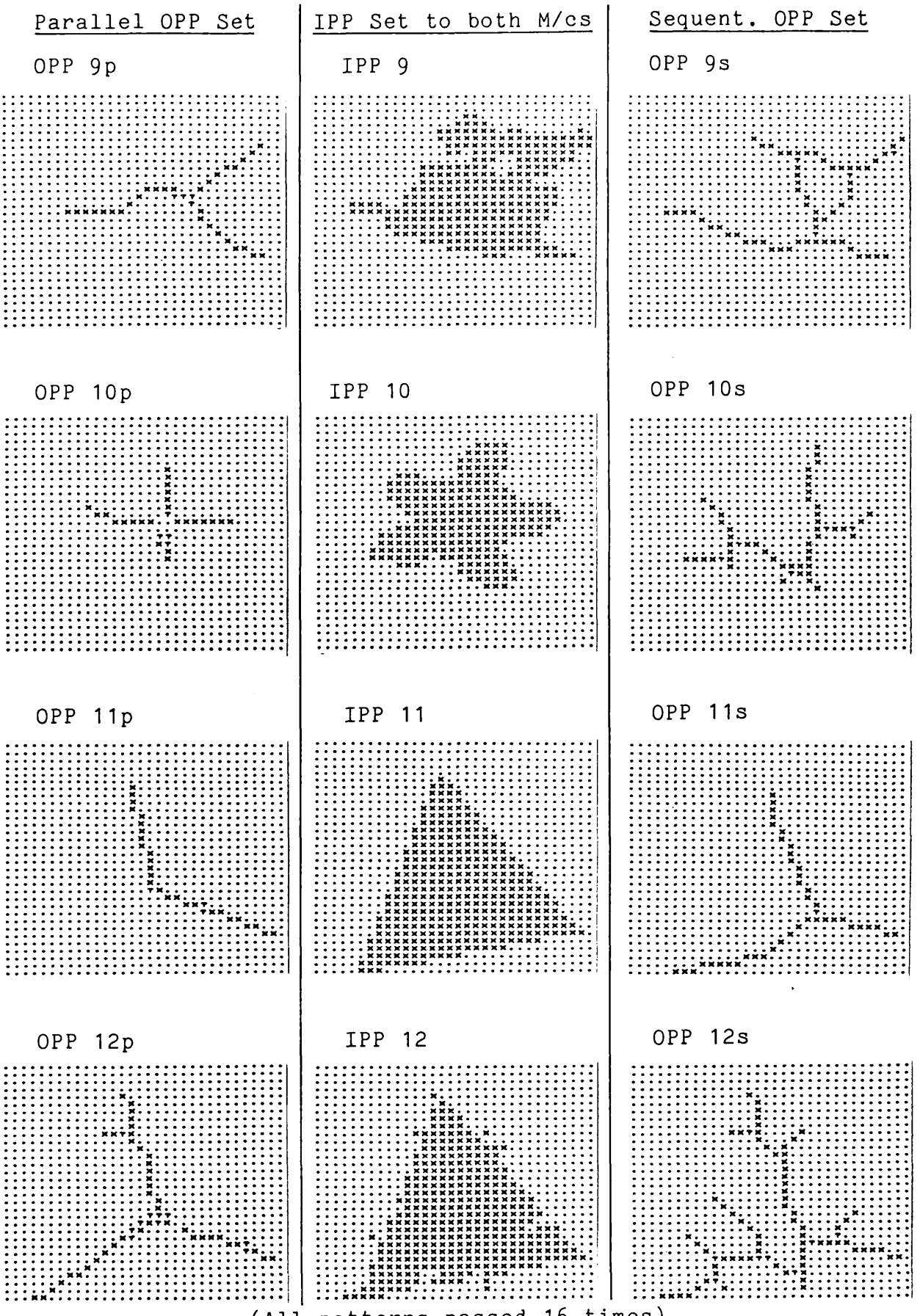
Experiment

A LPP machine was trained to 'thin and keep line ends' then applied in the sequential and parallel modes to a set of test inputs. These test inputs are shown in the centre columns of Figs 6.11 and 6.12. The results were produced using repeated feedback, the outputs being continually passed back into the machine until essentially static results were obtained. The OPPs produced using the sequential mode are reproduced on the right of Figs 6.11 and 6.12, and the parallel mode results on the left.



(All patterns passed 16 times)

Fig 6.11 Exp 13f : Comparison of Parallel/Sequential Applications with Feedback to IPPs 5-8



(All patterns passed 16 times)

Fig 6.12 Exp 13f : Comparison of Parallel/Sequential Applications with Feedback to IPPs 9-12

Results

The comparison of the results confirms the LPP machine behaviour as summarized in Fig 6.9 above. These characteristic differences are already well-known (20) and are not only a consideration restricted to trainable systems such as these.

Conclusion of Experiment 13

This concludes Experiments 13a to 13f, which have investigated the use of specially prepared training characters. This has shown how training may be adjusted to be quite specific in the tasks required (for example, 'keep or lose line ends') and can be improved to such an extent that other, more subtle effects on the test performance may be investigated. These investigations may also be further aided by the use of feedback in testing, to accentuate the processing behaviour. These machines have also been shown to exhibit the usual effects of parallel and sequential applications to test inputs.

6.2 Experiment 14 : Variations in the Addressing Function

In the previous experiments, the Address Calculator Function, as represented by 'f_2' in Section 4.2, has remained a simple binary word re-formatting function. That is, this function, which acts on a binary input W to form a binary output A ($A = f_2(W)$), does not perform any calculation or processing to generate its result. A processing function will now be attempted.

Consideration of the ways in which a binary input window may be transformed into a binary address leads to an

infinite variety of possible functions. These more sophisticated functions fall broadly into two groups - those that reflect the 'brightness' of the window, and those that are dependent on the 'structure' of the pattern contained within the window. (The function used earlier was more rudimentary, to the extent that it did not rely implicitly on either of the above characteristics of the window to generate the address.)

An example of a function of the former type would be a total or average value of black (or white) pixels in the window. This is obviously closely related to the brightness, but independent of the structure within the window. A function dependent on the structure might represent the concavity, convexity or limbs in the window pattern.

An address calculator function that relies on such features will be considerably more costly in processing time and complexity than the earlier simple re-stacking. However, it is likely to be more efficient in the size of the memory matrix required, as there is a necessarily smaller combination of characteristic features possible than the maximum possible number of windows (512) used earlier.

The F7 Format

The retention of a small (3x3) bit window extractor function results in a very small number of possible features of the type discussed above. Consequently, a single new address calculator function was created that made use of both types of characteristics - brightness and structure.

Consider the ring of eight-connected neighbours around the window centre point P0. A function that relies on the 'brightness' of such a ring is the simple total number of

pixels ('t') set equal to '1'. Here, this value 't' may range from 0 to 8.

A function that depends on the structure within this window may be generated by calculation of the crossing number around the ring of immediate neighbours of the centre point. The crossing number (X) is defined as the number of transitions between black and white as the pixels around the centre point are examined in a ring. 'X' is necessarily an even number, and hence the value 'k' (=X/2) will be used to avoid redundancy. Clearly, k is the number of limbs connecting to the centre point of the window and may range from 0 to 4 on this rectangular lattice.

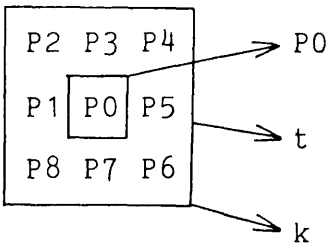
Dependence on this centre point value P0 is retained as a single bit in the address. This address calculator function is thus composed of these three characteristics, the values 'k,t,P0' arranged as in Fig 6.13 below.

(This change in the LPP machine format is effected by a single subroutine change, as described later in Section 7.5. The original subroutine 'CADDR' which corresponds to the 'Address Calculator (f_2)' is listed in Appendix 1a (lines 5000 to 5510 for the F5 LPP machine format. The new version of 'CADDR' for this F7 format simply replaces the original subroutine.)

Experiment

The F7 LPP machine as described above was set up in the software simulator and trained on the pair of specially prepared patterns used in the previous experiment. That is, the pair of patterns shown in Fig 6.1b was used, designed to illustrate the task :

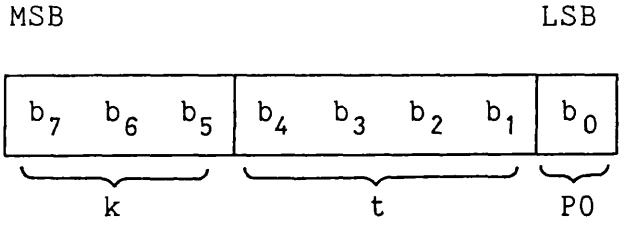
9-bit I/P Window : W



P0	1-bit	Centre point pixel Two values : 0 or 1
t	4-bit	Total pixel value in ring Nine values : 0 to 8
k	3-bit	Half-crossing number in ring Five values : 0 to 4

These are combined to form :

8-bit Memory Matrix Address : A



Hence M.M in practice uses 256 ($=2^8$) cells,

although less than 90 (=2x9x5) are accessible.

Fig 6.13 F7 Machine Address Calculator Function

'thin, maintain connectedness and keep line ends.'

This use of a training set known to be effective was to focus the investigation on the changes in performance due to the changes in machine format alone - to avoid the problem of poor training masking this effect. Consequently, the same test IPPs, modes of operation (parallel and then sequential) and number of feedback passes (ie. 16) were all used here, as in part of the previous Experiment 13f shown in Fig 6.11. The four test IPPs and the corresponding OPPs generated by the parallel and sequential modes (all with feedback applied

to ensure completion of the process) are shown in Fig 6.14. These are to be compared directly with the F5 machine results in Fig 6.11.

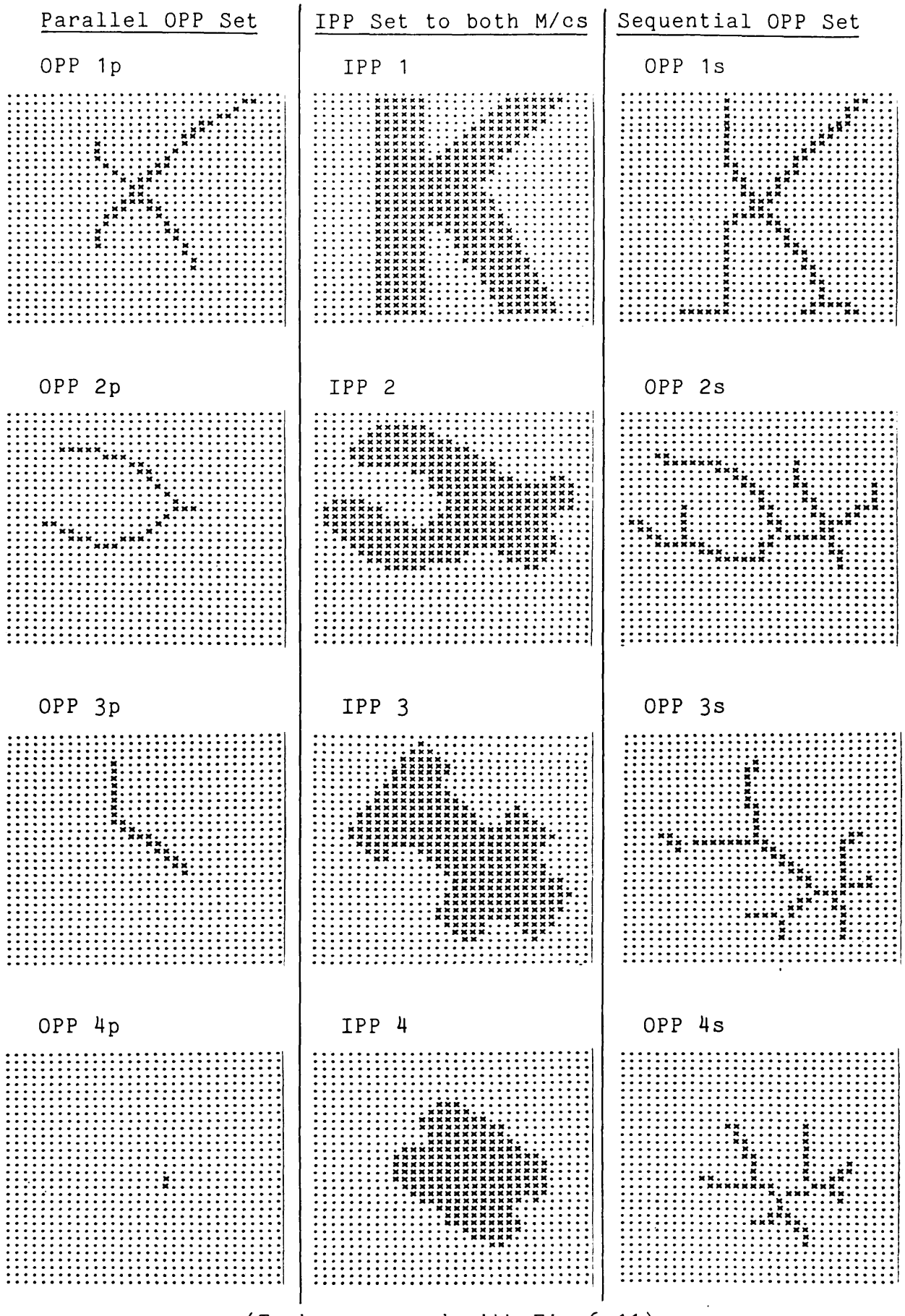
Results

The results obtained here with the F7 machine are broadly similar to those obtained earlier with the F5 machine. That is, the IPPs have all been thinned to a unit width skeleton on repeated application of the thinning algorithm. The expected variation in performance seen between parallel and sequential processing has occurred again, further confirming the earlier analysis of this behaviour.

However, there are some interesting non-random differences between the results of the F5 and F7 machines. The skeletons produced by the F5 machine were single width and generally eight-connected. The corresponding single width limbs from the F7 machine are four-connected, the difference being especially apparent in diagonal lines. This difference is illustrated below in Fig 6.15.

It will also be noted that there are no '?' pixels, as the machine was preset to contain '1' pixels. This is necessary for the reasons explained in the discussion below. (The alternative of presetting the machine to '0' pixels before training makes little difference to the outputs.)

The results with the parallel mode of operation produced static results after 16 feedback passes, whereas the sequential mode of operation produced results that were often in a small local state of stable oscillation. That is, on each pass a local transformation on a particular feature produced an output that was eventually transformed back into



(To be compared with Fig 6.11)

Fig 6.14 Experiment 14 : Test Results

Single Width Connected Limbs :

```
. . . . . . . . . . . . .
. . . . . . . . . . . . .
X X X X X . . . . . . . .
. . . . . X . . . . . . .
. . . . . . X . . . . . .
. . . . . . . X . . . . .
. . . . . . . . X . . . .
. . . . . . . . . X . . .
. . . . . . . . . X . . .
. . . . . . . . . X . . .
. . . . . . . . . X . . .
. . . . . . . . . X . . .
```

Eight-connected
Skeletal Limbs

(Similar to those in
Fig 6.11 : F5 m/c.)

```
. . . . . . . . . . . . .
. . . . . . . . . . . . .
X X X X X . . . . . . . .
. . . . . X X . . . . . .
. . . . . . X X . . . . .
. . . . . . . X X . . . .
. . . . . . . . X X . . .
. . . . . . . . . X X . .
. . . . . . . . . . X . .
. . . . . . . . . . X . .
. . . . . . . . . . X . .
. . . . . . . . . . X . .
```

Four-connected
Skeletal Limbs

(Similar to those in
Fig 6.14 : F7 m/c.)

Fig 6.15 Differences Between Eight and Four-connected Limbs as Exhibited by F5 and F7 Machines

the original feature. The particular cycle that occurred is illustrated in Fig 6.16 below. These errors are due to inadequate training, where such patterns have not occurred in the training set. Unlike the F5 machine, the F7 machine must necessarily output definite values on each pass, hence the possibility of oscillatory behaviour.

```
X . . . .    X . . . .    X . . . .    X . . . .
. X . . .    . X . . .    . X . . .    . X . . .
. X X X X    . . X X X    . . . X X    . X X X X    etc...
. X . . .    . X . . .    . X . . .    . X . . .
X . . . .    X . . . .    X . . . .    X . . . .
```

F7 Machine (trained on patterns in Fig 6.1b) produces this cyclic behaviour on this feature on each successive feedback pass in the sequential mode of operation.

Fig 6.16 Local Stable Oscillatory Behaviour

Discussion

It is difficult to relate the reasons for the differences in results to the structure of the F5 and F7 machines. The more complex address calculator function (F7) means that not only is intuitive analysis difficult, but so is an analysis based on statistical examination of the machine's internal state. Neither an analysis of the TP value (derived from examination of cell totals) nor a physical listing of the memory matrix cells (as in Appendix 2 - albeit in a different format) is of much interpretive use.

However, it is interesting to note that such a machine, with this complex addressing function, not only works but works relatively well. Certainly the test performance compared with the F5 machine reveals similar picture processing ability having been trained by the same experiences only. It should also be noted that the memory matrix size has been reduced from 512 to less than 90 cells, yet comparable results have been produced.

(Less than 90 cells are actually used for the following reason. Although the three variables (k,t,P0) may take 5, 9 and 2 values respectively, suggesting their product as the total number of cells required, several cells are never used. This is because they represent impossible combinations. For example, it is obvious that t and 8-t must be greater than or equal to k for consistency, resulting in several cells' addresses never arising, and the corresponding use of considerably less than 90 cells in the memory matrix.)

The major repeatable difference in testing is seen as a variation in dealing with the structure of the pattern within the 3x3 window. This is not unexpected as the F7 machine relies on this structure in a totally different manner from the F5 machine, and hence reacts differently in testing. This sensitivity to structure is in some sense an improvement in performance.

There are three further points to discuss concerning this specific machine, which arise from theoretical considerations rather than experimental results.

1 Rotation and Reflection

The more complex address calculator function generating k and t effectively replaces the requirement used in previous formats to rotate and reflect the input windows in training. These new cell addresses are invariant under rotation and reflection and thus implement this function automatically. Consequently, in building this new format in the simulator, not only was the 'CADDR' subroutine changed as described above, but also the routine to rotate and reflect the input window was removed from the software.

2 Memory Matrix Preset Value

The value to which the memory matrix cells are preset before training must be examined closely in conjunction with the mode of operation in which the machine will be run. Specifically, the preset value must depend on whether or not the machine will be tested with feedback. This is because if the memory matrix cells are preset to '?' as before, the subsequent feedback of OPPs containing '?' pixels cannot be easily resolved into a consistent value for k or t. These

totals ideally should be generated from binary (not trinary) variables. The conflict occurs in the former case (k) in deciding whether :

'?' pixels	=	'0' pixels	or	'1' pixels

in the comparison to define the changes of state as the ring of neighbours is examined, and in the latter case (t) deciding whether '?' pixels should be regarded as 'black', 'white' or something in between. This problem was resolved in this experiment by presetting the cells in the memory matrix to contain '1' pixels. (The alternative of presetting the cells to '0' made little experimental difference.)

3 Memory Matrix Effective Size

The effective size of the memory matrix of the F7 machine has been reduced from 512 to less than 90 cells, compared with a reduction from 512 to 102 with the F5 machine. While this reduction is comparable in this case, much larger reductions could be envisaged when larger windows are used with an F7-like machine in future.

Interpretations of Variations in the Addressing Format

The above experiment varied the memory matrix addressing format used from the earlier simple 9-bit pixel stack address. This leads to the question as to the theoretical optimum memory matrix addressing mode that may be generated from a 9-bit binary input window.

The address calculator function ($A=f_2(W)$) that takes a 9-bit argument and simply re-formats this to a 9-bit binary result loses no information at all in this process. That is, the F5 machine's memory matrix must be potentially

the most capacious in terms of the ultimate information storage. However, this is obviously to be weighed against the more efficient alternatives that may store the same information in less space. The F7 format would seem to be such a candidate from the results of these experimental observations. There is an added complication to be weighed in the search for the optimum performance - the computational cost of such a calculating function. This acts against the information storage efficiency, and again, each potential system must be evaluated with this in mind.

Unfortunately, in this imprecise field of training a picture processor by examples (which themselves must be imprecise if they are to represent practical data, gathered from realistic situations) the only method of ascertaining the performance is by pragmatic examination of test results. This too will be imprecise, but in many such complex systems this is often the most efficient method of searching for a justifiable conclusion on performance.

Other Window Formats

This change in the addressing format illustrates how a more memory efficient machine may be produced which is still capable of processing pictures when viewed through a 3x3-bit window. Changes in the window itself are also possible; that is, in the Input or Example Window Extractors as represented in Section 4.2 as :

$$W = f_1(IPP , x , y) \qquad \text{Input Window Extractor}$$

$$EXP/PO = f_4(EXP , x , y) \qquad \text{Example Window Extractor}$$

Examples of variations in the input window can be simple extensions of the size (beyond 3x3, or even below 3x3

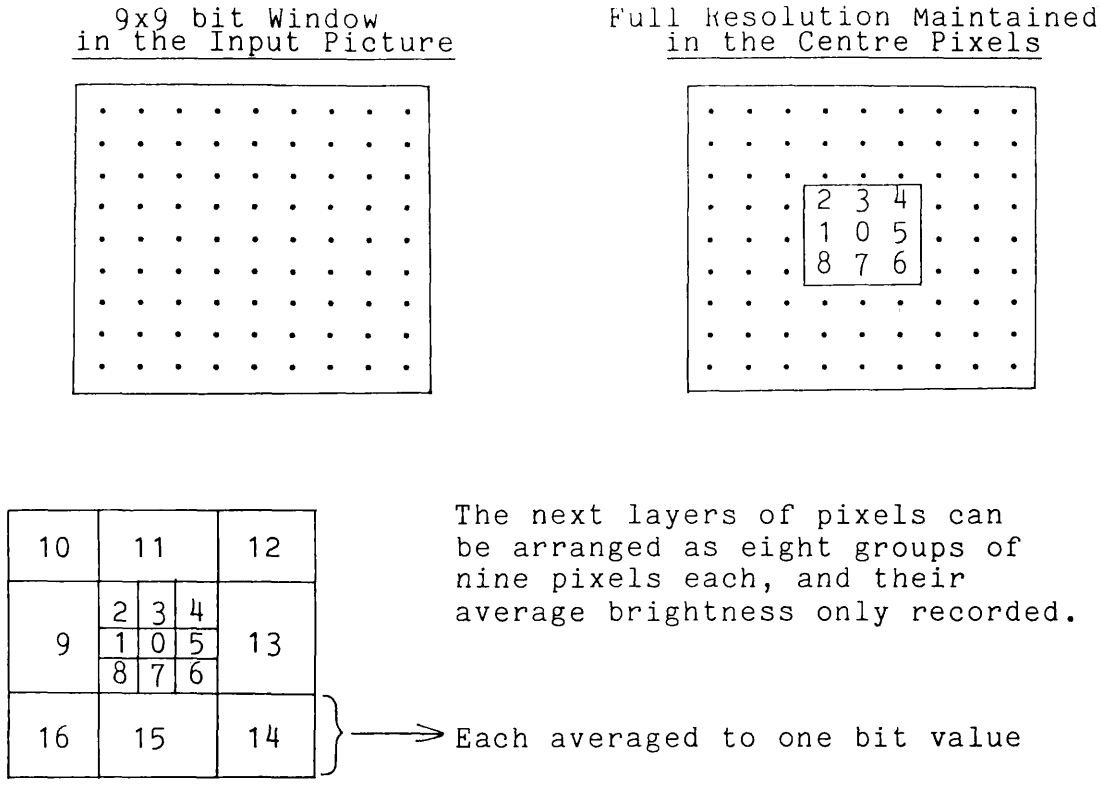
as in Experiment 10) to encompass a feature size more closely suited to the type of processing envisaged. The resolution of pixels need not be constant over the window, if such a system were found efficient. An example of this type is illustrated in Fig 6.17 below, where a reduction in resolution away from the centre point results in a 'local pyramid' type of configuration. This use of such pyramids over a local window is an extension of the already established use of such quadtrees and pyramidal data structures as picture descriptors over the entire field (5,27,76). Here, within a local window, the full pixel resolution is retained at the centre point, yet the next three layers of pixels are 'averaged' to generate a single-bit brightness value for each of the eight surrounding groups of pixels.

Such a window format would take detailed account of the centre point and its immediate neighbours, and depend less on a broader spread of pixels. This may be a better compromise than an equivalent system where full resolution is retained over the entire input window.

This system could be extended indefinitely, reducing resolution as more neighbours are encompassed. It is suggested here as a possibly more efficient method of processing local windows by such trainable systems; but has not been attempted experimentally in this work.

6.3 Experiment 15 : Locating and Tracking of Objects

A short experiment in this series was performed to examine how a trainable picture processing system would



This results in a 17 bit window with resolution high near centre point and lower as distance from centre increases.

Fig 6.17 An Example of a Pyramidal Type of Representation As a Variation in Window Format

tackle a task somewhat different from the 'shape analysis' range attempted above. This task was the location and tracking of objects within the field of view. Location of objects by such a machine could be accomplished by simply marking the object desired in the field, and tracking could possibly move or modify the desired object in the desired manner. The location task was attempted first.

Experiment

The training of a F5 machine was accomplished using a field containing a large object among several smaller objects as IPP (to represent 'a target against a noisy

background' for example). The EXP was a lone marker centred on the larger object, thus defining the task as 'produce a marker on the desired object'. A set of eight such hand-drawn pairs were used to train the F5 LPP machine, three of which are illustrated in Fig 6.18. Examination of these patterns will show clearly the task attempted.

The memory matrix contents, as they changed throughout the training period were 'totalized' after each training pair was applied, and the TP value at each stage calculated. These data are shown in Fig 6.19 as a table and histogram.

The machine was tested in parallel on a set of input patterns shown in the left hand column of Fig 6.20. These can be seen to represent such 'target' pictures with an increasingly noisy background. The resultant set of OPPs are shown on the right of this figure.

Results

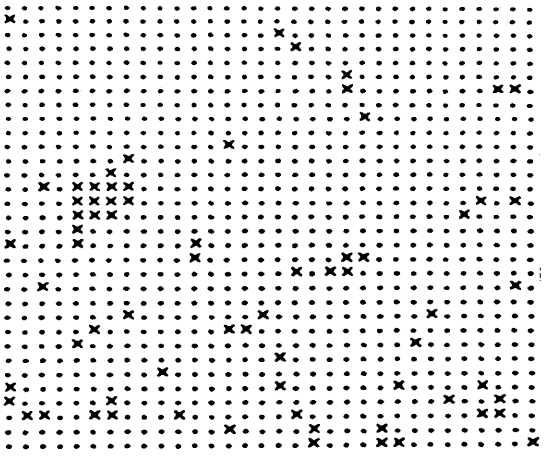
The test inputs with little or no noise are correctly processed to give a single marker appearing in the output centred on the object to be located. As the noise in the input increases, the machine eventually starts to react to this by generating marker outputs at the position of the larger noise objects in addition to the required object.

Discussion

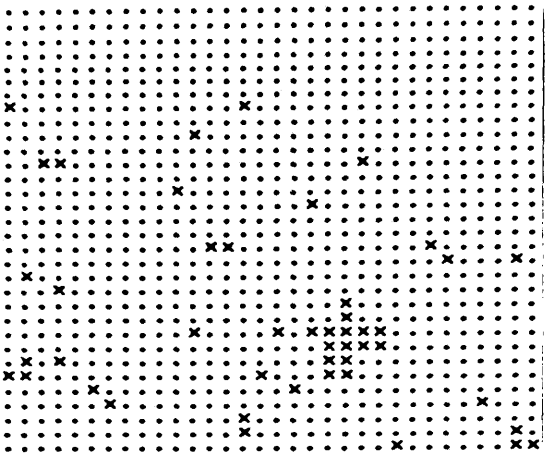
The ability to locate objects correctly in the face of moderate noise indicates successful operation of the LPP machine in these cases. The greater noise producing spurious results is to be expected, in that the machine is obviously reacting to the size of objects only, not any other intrinsic properties such as shape or orientation.

Training IPPs

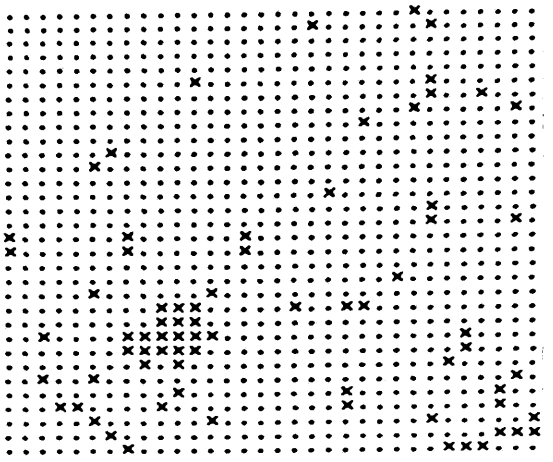
IPP 1



IPP 2

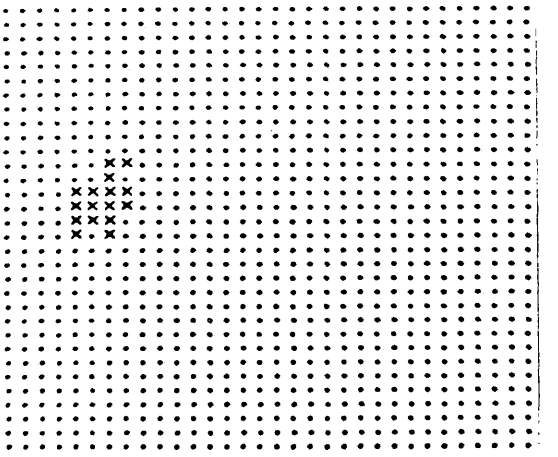


IPP 3

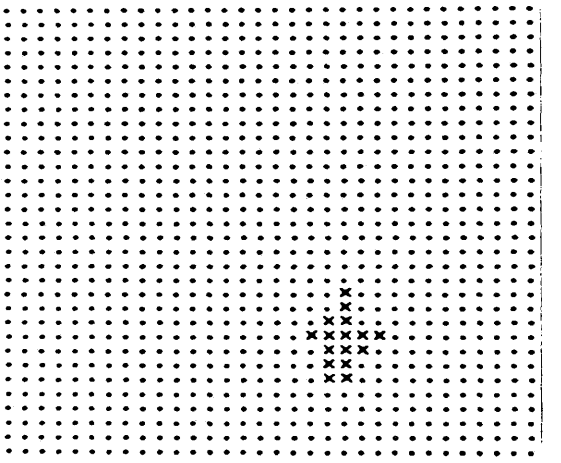


Training EXPs

EXP 1



EXP 2



EXP 3

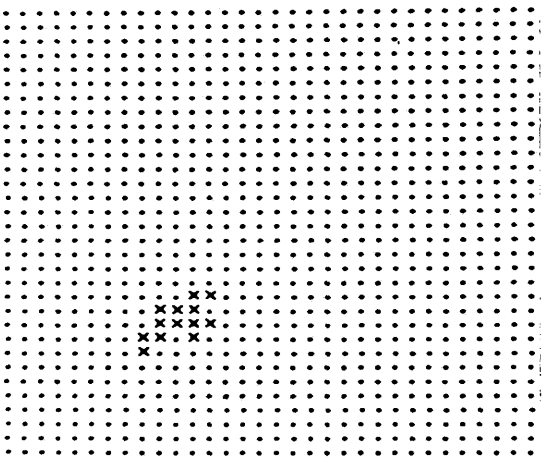


Fig 6.18 Experiment 15 : 3 of 8 Pairs from Training Set

No. of Training Patterns Received	No. of Cells=0	No. of Cells=?	No. of Cells=1	No. of Trained Cells
0	0 (0%)	512 (100%)	0 (0%)	0 (0%)
1	142 (27.8%)	313 (61.1%)	57 (11.1%)	199 (38.9%)
2	158 (30.8%)	285 (55.7%)	69 (13.5%)	227 (44.3%)
3	207 (40.4%)	249 (48.6%)	56 (11.0%)	263 (51.4%)
4	238 (46.5%)	215 (42.0%)	59 (11.5%)	297 (58.0%)
5	230 (44.9%)	191 (37.3%)	91 (17.8%)	321 (62.7%)
6	234 (45.7%)	183 (35.7%)	95 (18.6%)	329 (64.3%)
7	246 (48.1%)	163 (31.8%)	103 (20.1%)	349 (68.2%)
8	234 (45.7%)	147 (28.7%)	131 (25.6%)	365 (71.3%)

Histogram of Above Data : (Key as in Fig 5.9)

No. of Patterns :

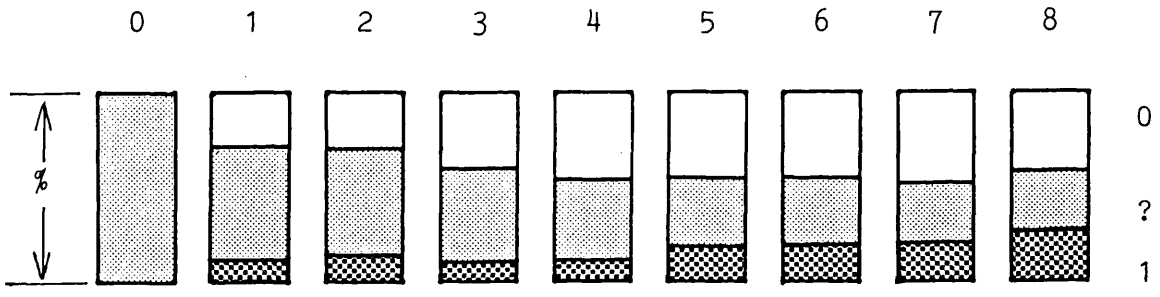


Fig 6.19 Exp 15 : Development of M.M Contents During Training Phase

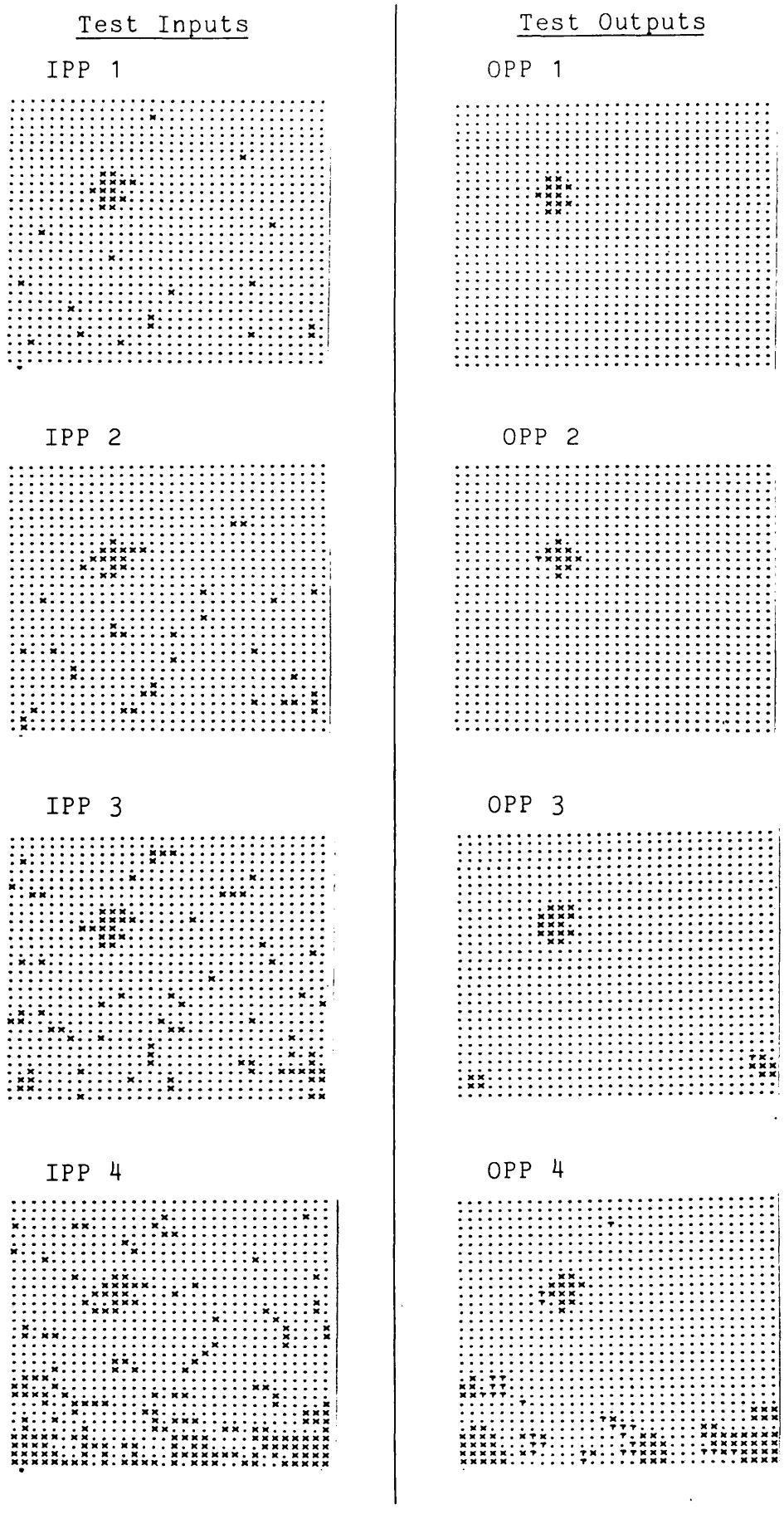


Fig 6.20 Experiment 15 : Test Results

Thus, the test results confirm the training was as thorough as is possible with such a machine; indeed, the TP value obtained of 71.3% (see Fig 6.19) confirms that the quantity of training was ample.

For the machine to be able to differentiate between the desired object and noise (or any other objects) it must be trained on information that is 'visible' through the (in this case) 3x3 bit window. Such a small window used here precludes the use of any intricate structure or design in the required object as a distinguishing feature to which the machine may react. Consequently, 'size' compared to the 3x3 window is the only feature available to the machine trained in this manner, thus accounting for its behaviour.

This determination of size using a small window is encouraging, given the usual limitations of such windows. However, this size is strictly limited to that directly comparable with the window.

The task of 'tracking' (in which an object may have its motion on two temporally successive frames predicted) is frustrated by this small window also. (This has been verified by experiments not described fully here, where the outputs showed no coherent response.) It would be necessary to distinguish between the 'front' (advancing) and 'back' (receding) edges of the required object for the machine to be able to predict the next frame showing the direction of motion of the object. Such distinguishable features are just conceivable on a 3x3 window, but these would not represent realistic, distorted features, such as would be received 'off-camera' in any real situation.

The Need for Bigger Windows

The limit of possible processing in this task (and others) with this small window has been reached. Further investigations must make use of larger windows - large enough to enable detailed structure of pictorial objects to be contained within a single window. The use of a larger window should also be accompanied by the corresponding increase in overall frame size (that is, greater than 32^2) to enable a field of view much larger than the window extracted. The combination of these two requirements may be stated alternatively by stipulating that a far greater overall resolution or finesse should be used in defining discrete pixels in the original digitisation process.

6.4 Experiments 16 and 17 : LPP Machine 'Neurosis'

There is a need to develop a more generalised approach to understanding these LPP machines. Experiments have been performed that give performances of specific machines in specific applications. It has emerged that this performance is predominantly dependent on the training received, and to a lesser extent on the machine format and operation.

It is wished to shift the experimental emphasis from the investigation of particular cases to the development of a more useful tool in assessing the potential for picture processing by learning machines. The quality of training is known to be paramount, and so a more penetrating analysis of this quality is highly desirable. This will not necessarily lead to better ultimate performance, but certainly to a better understanding of the capabilities of these machines.

An LPP Machine sensitive to Neurosis : the F8 Machine

As it happens, a relatively simple variant of the LPP machines already developed gives much of this type of information, previously lost. The 'F8' machine was created, which is capable of recording a 'neurotic' memory matrix cell - one that has been trained to different values on different occasions - ie. inconsistently trained. This is done by introducing a fourth possible state for the cells. The state transition diagrams for responses to the various stimuli are shown in Fig 6.21 to explain this. A comparison is made with the most similar F5 machine.

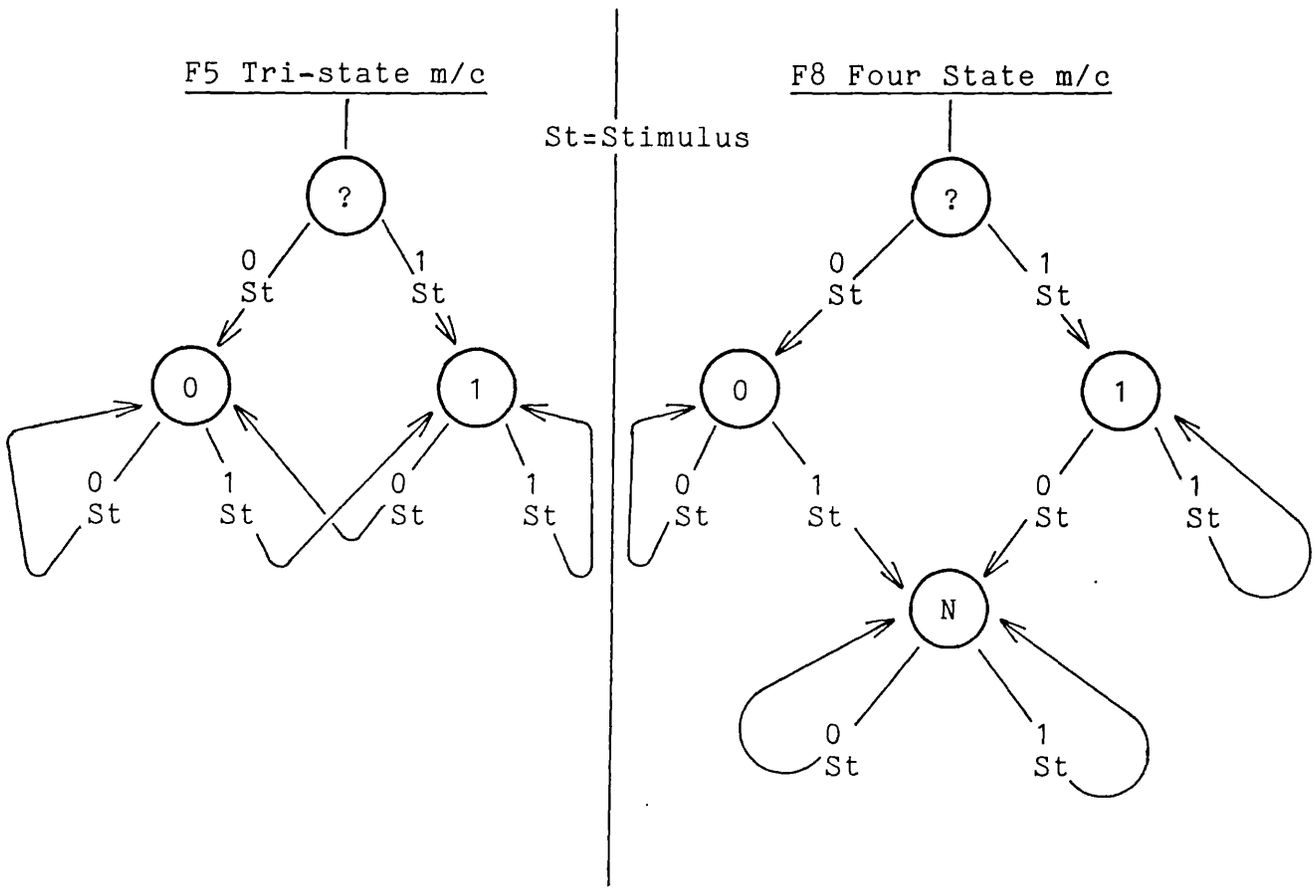


Fig 6.21 State Transition Diagrams for F5 and F8 Machines

It can be seen that any attempt at inconsistent training with the F8 machine will result in a cell being set

to 'N' permanently.

Experiment 16

As a practical exercise for this F8 machine, it was trained on the second training set used in Experiment 13 (see Fig 6.1b). The results (which now show the extent of 'neurotic' memory after training) are shown in Fig 6.22, again compared with the original results from Experiment 13 using the F5 machine.

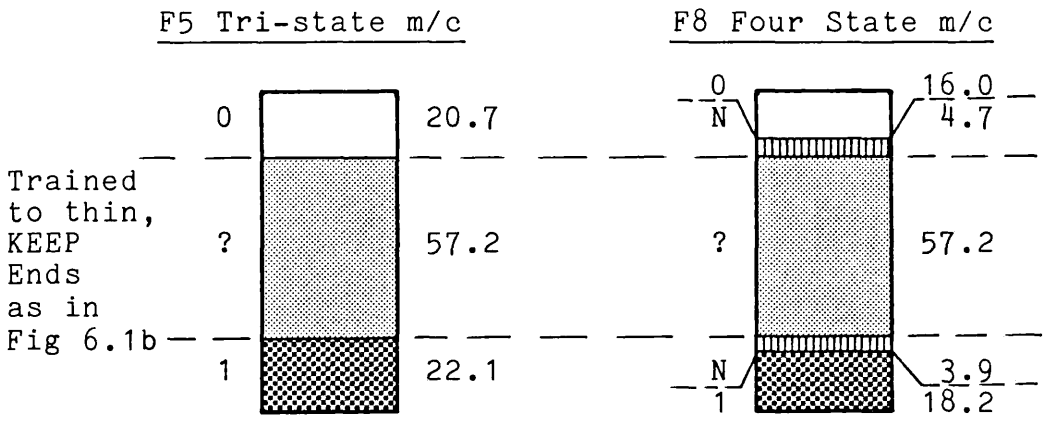


Fig 6.22 Exp 16 : Memory Matrices of F5 and F8 Machines

(Although only one type of 'N' cell is detectable after training, comparison of the F5 and F8 results allow the separation of the 'N' cells into two bands - those that were ultimately left equal to '0' in the F5 machine, and those left equal to '1'. Hence the two bands of 'N' cells in the F8 machine result.)

Results

The use of the F8 machine shows a result of 8.6% neurotic cells after training. This must be classed as

'good' training, as the remainder of Experiment 13 showed to be the case.

Discussion

This short demonstration serves to illustrate the potential now available for assessing the quality of training. This is a considerably more powerful measure than the TP value developed earlier, which merely measured the quantity of training, both consistent and inconsistent.

Before a full discussion of the implications of this type of investigation, a further short development is made below, generating an even more powerful tool for LPP machine behaviour analysis.

The F9 Machine

A final variant (F9) machine was set up with the following response in each memory matrix cell to training stimuli. Each cell contains two counters, one to count the number of '0' stimuli received, the other to count the number of '1' stimuli. Both counters are initially set to zero before training and saturate if they reach their maximum value. (In this implementation, these are two four-bit binary counters, and thus each has a range of 0-15.)

Experiment 17

The F9 machine was again trained on the set used in Experiment 13 (Fig 6.1b). A second training session was also implemented to provide further data for investigation. This training used the set in Experiment 15, three of the eight pairs used being shown in Fig 6.18.

Results

The raw results (the two memory matrices after training) are not reproduced here, as they require some manipulation to become informative. Similarly, there are no test picture results, as no algorithm for using these matrices in testing has been defined. (The investigation is concerned solely with the training and the machine's reaction to this.)

Discussion

Sixteen values may be recorded in either of the two counters in each memory matrix cell. To present the information usefully, the total number of cells left in each of the 256 possible combinations are counted. This is shown in Fig 6.23 below, where a table of 16 rows and 16 columns for each memory matrix is shown. In each entry in the table, the number of cells with a '0' total and a '1' total given by the axes is shown.

There are several points to note :

1 The left hand side of the table corresponds to few '0' stimuli, the right hand side to many '0' stimuli,

2 Similarly, the lower region corresponds to few '1' stimuli, the upper region to many.

This results in :

3 Those totals grouped around the lower left hand corner correspond to little training of any kind. (Indeed, those in the 0,0 square correspond to completely untrained cells, '?' when implemented

(a) Trained on Set in Fig 6.1b

'1' Stimuli Rec'd	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15+
15+	5	.	.	.	.	.	.	.	.	.	.	.	.	.	8	.
14	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
13	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
12	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
11	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
10	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
9	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	4
8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
6	8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
4	10	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
3	24	.	8	.	.	.	.	.	.	.	.	.	.	.	.	.
2	20	8	4	.	.	.	.	.	.	.	.	.	.	.	.	.
1	26	.	.	.	.	.	.	.	.	.	.	.	.	.	4	8
0	293	17	4	.	12	12	.	4	.	.	.	.	.	4	.	29

'0' Stimuli Rec'd →

(b) Trained on Set in Fig 6.18

'1' Stimuli Rec'd	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15+
15+	.	.	8	.	.	.	.	.	.	.	.	.	.	.	.	.
14	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.
13	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
12	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
11	.	.	.	.	.	3	.	.	.	.	.	.	.	.	.	.
10	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
9	4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7	4	4	.	8	.	.	.	.	.	.	.	.	.	.	.	.
6	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.
5	.	8	.	.	.	.	.	.	.	.	.	.	.	.	.	8
4	.	8	.	.	.	.	.	.	.	.	.	.	.	.	.	4
3	12	4	8	.	.	.	.	.	.	.	.	.	.	.	.	8
2	.	.	.	.	8	.	4	.	.	.	.	.	.	.	.	4
1	37	12	.	4	.	.	.	.	.	8	.	.	.	.	.	20
0	148	48	34	20	8	.	8	8	8	.	2	.	.	4	.	34

'0' Stimuli Rec'd →

Fig 6.23 Exp 17 : Totals for Two F9 M/c Memory Matrices

before.)

4 Those totals grouped in the upper left hand corner correspond to predominantly '1' stimuli (at least more '1's than '0's seen in training) and thus imply consistent training to generate '1' outputs. A similar situation applies for '0' stimuli in the lower right hand corner.

5 The upper right region represents cells that have seen considerable amounts of both '0' and '1' stimuli. This corresponds to inconsistent training.

Behaviour during Training

A set of stylized tables of this type as they would be expected to develop during training is given in Fig 6.24.

Decisions in Testing

Such a table can be used to experiment with different decision boundaries in the testing phase. Examples of possible decision boundaries and their implications are given in Fig 6.25. This gives insight into how a LPP machine may be much more finely 'tuned' in testing to take maximum advantage of a non-ideal training set presented to it.

Numerical Analysis

The totals in the rows and columns in these tables may themselves be totalised, yet in one very important sense this loses the very information we have been at pains to gather : the correlation between different '0' and '1' stimuli for the same feature. A more profitable analysis involves the totalling of the entries in the tables along the diagonals. There are two sets of diagonals and they both

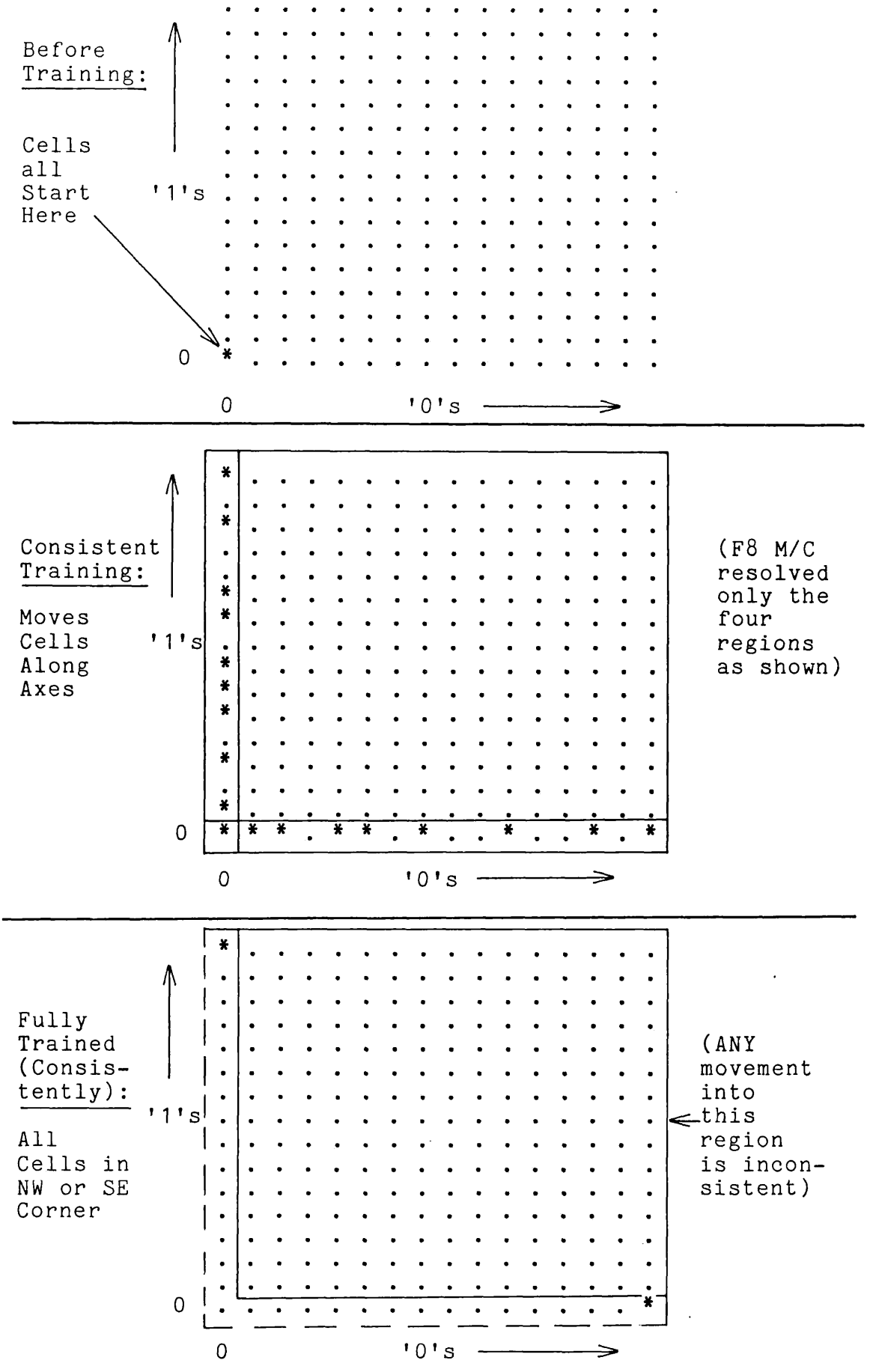
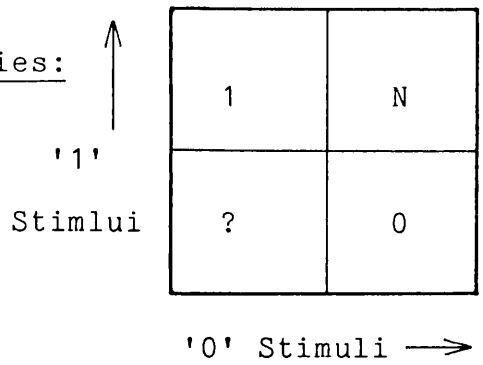


Fig 6.24 F9 M/c Development During Training

Simple Boundaries:

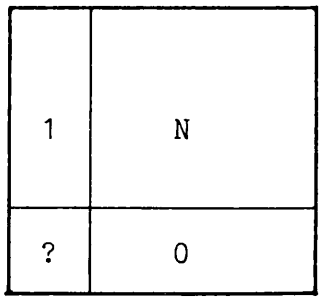


? - Output '?'
0 - Output '0'
1 - Output '1'
N - Output 'N'

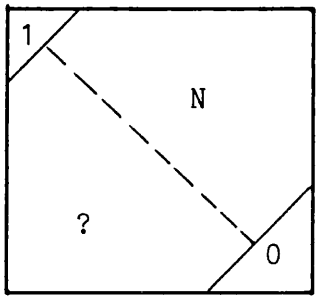
under test if cell lies in this region

Not necessarily equally sized regions :

Eg if risk of mis-processing is high :

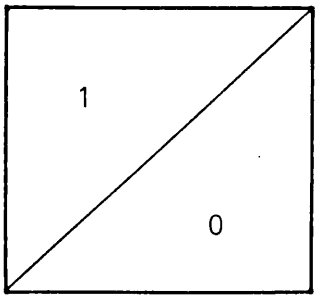


or :



(Possibly unnecessary to distinguish between '?' and 'N')

Alternatively, if a decision must be made, whatever the risk :



This is simply: 'Choose the output whose total is bigger'

Fig 6.25 F9 M/c : Testing Decision Boundaries

generate useful data as indicated in Fig 6.26. An illustration is also shown of the expected development of these tables during training.

This data has been generated and plotted in Fig 6.27, from the data in the tables of Fig 6.23. These give a numerical feel for the two most important considerations here when implementing a LPP machine :

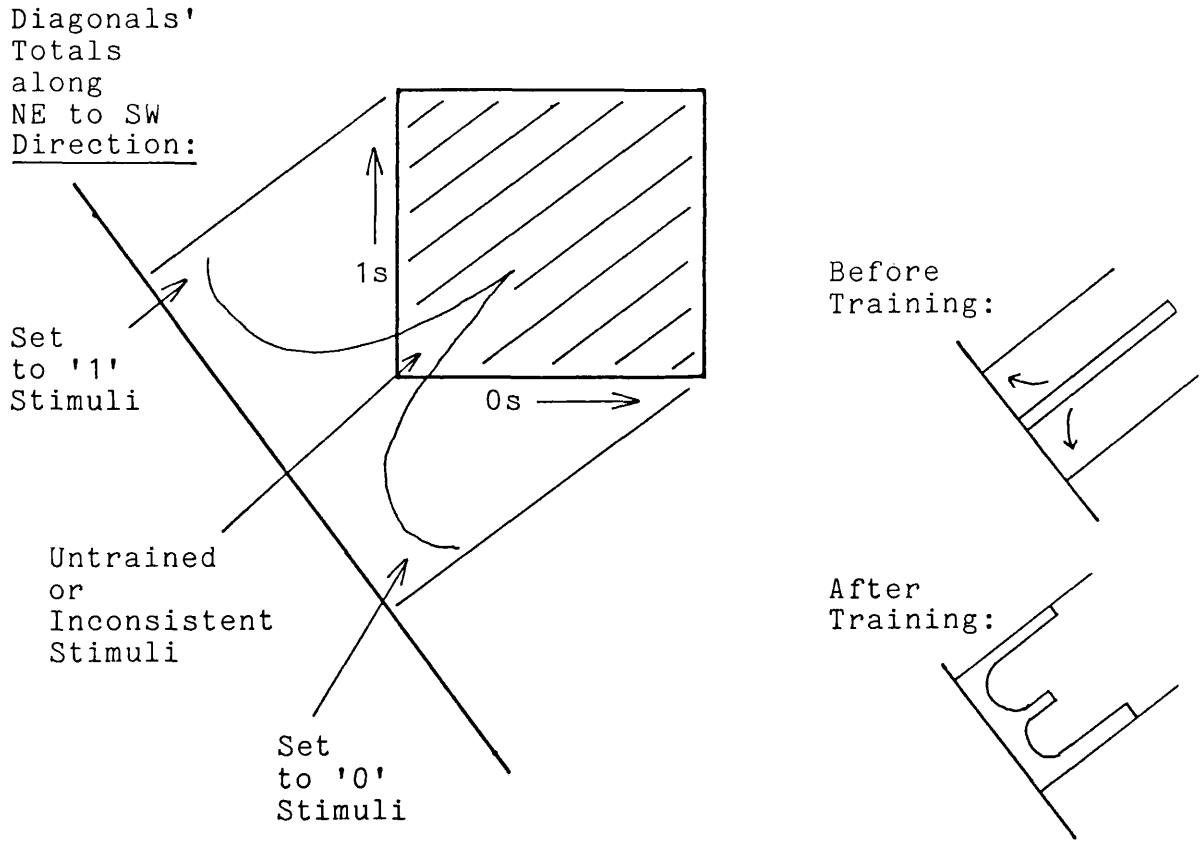
1 the extent to which the available training is consistent and unlikely to confuse the machine,

2 the minimum size of machine needed to resolve easily between the three peaks of insufficient, accurate and inconsistent training.

The first consideration above would draw information from the second and fourth graphs in Fig 6.27 (ie. 'b' and 'd'), the diagonal totals taken along a NW to SE direction. This shows the three peaks of insufficient, consistent and inconsistent training which could be broadly separated by the suggested boundaries shown dotted on the graphs.

The first training set (used to generate the Fig 6.27b graph) can be seen to contain small, but readily identifiable peaks of inconsistent points, which as such can be guarded against with the F9 machine. Similarly, there is a clear 'ground' between the peaks of insufficient and consistent training in both graphs 'b' and 'd'; where a decision boundary can be confidently drawn.

The second consideration above uses the graphs 'a' and 'c' to measure how the available machine resolution compares with the training data, and to draw the ultimate decision boundaries to be used in testing. These translate to



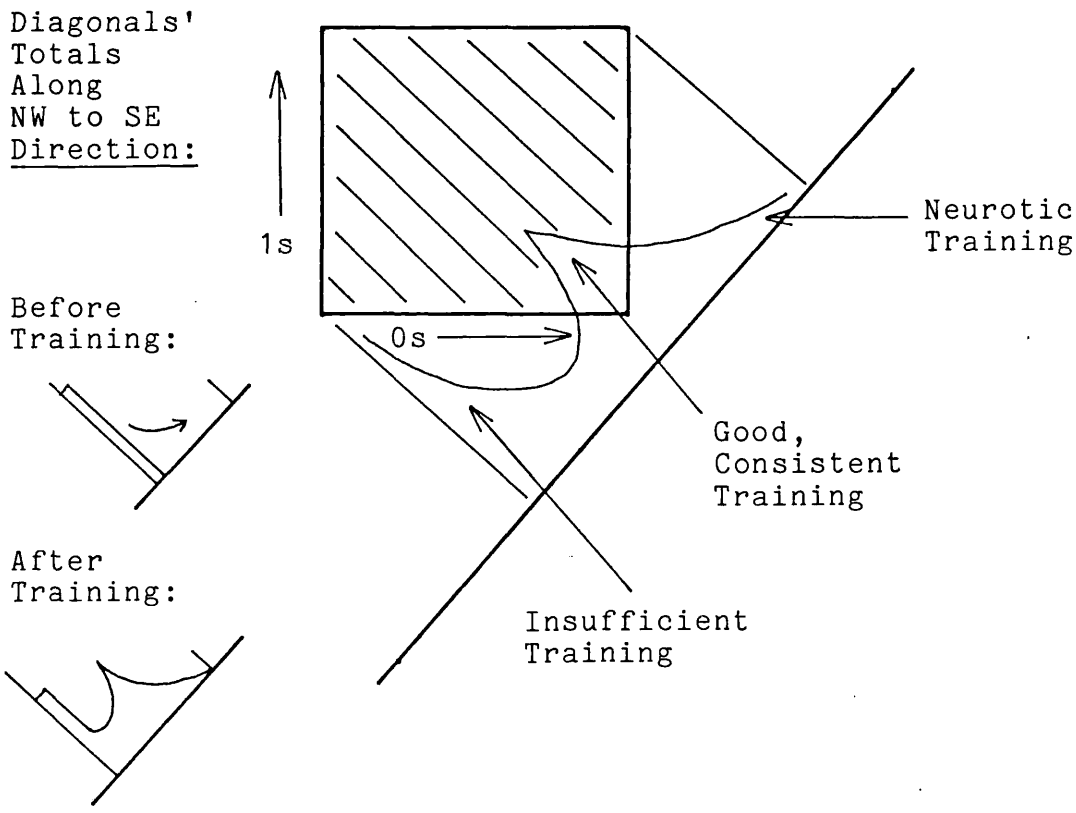


Fig 6.26 F9 M/c : Implications of Diagonals' Totals

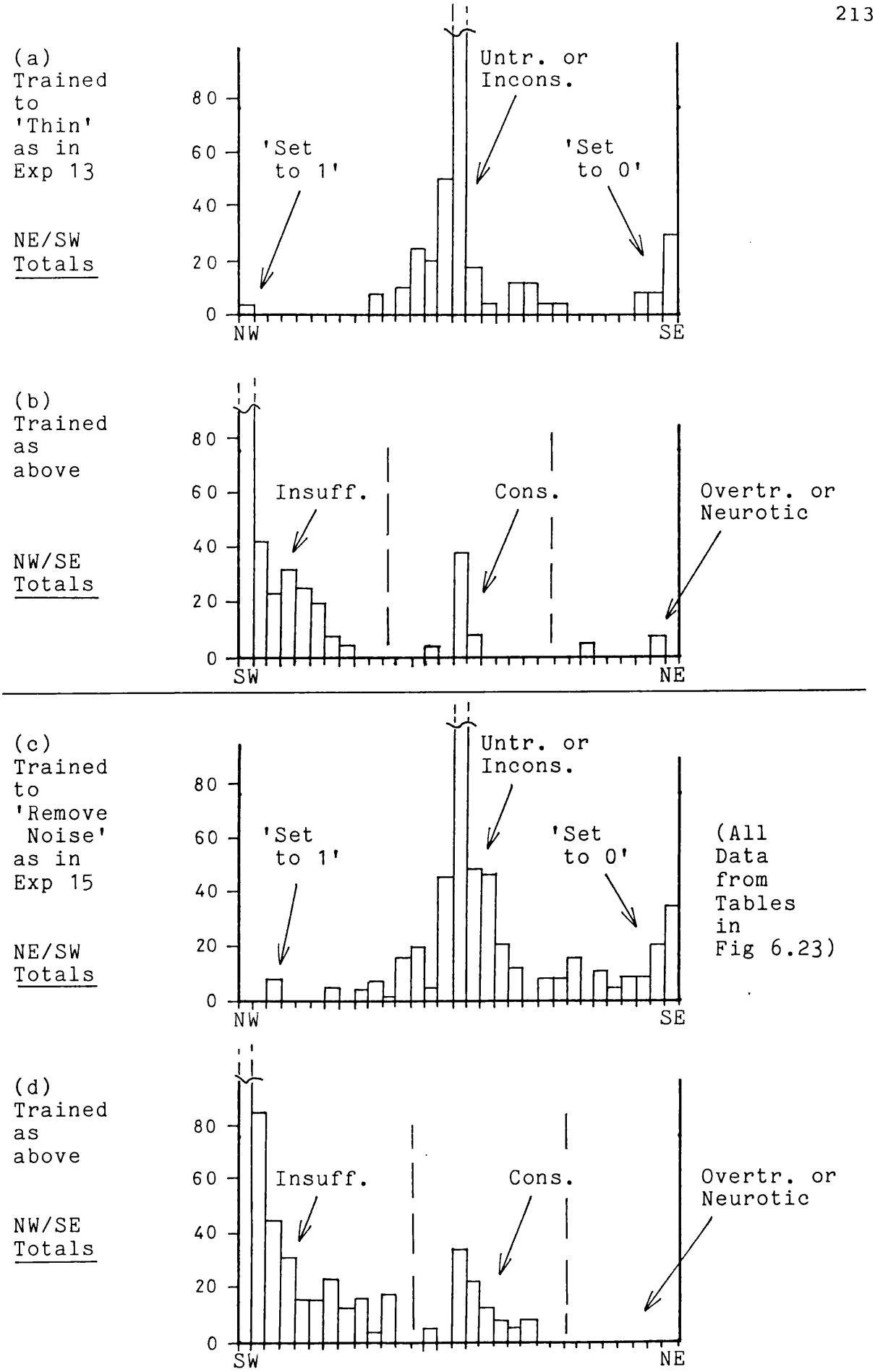


Fig 6.27 Exp 17 : Diagonals' Totals Graphs

diagonal boundaries on the tables in Fig 6.23, in the manner shown in Fig 6.25. In the case of graph 'a', the machine is obviously capable of easily resolving the peaks of '1', 'untrained or inconsistent' and '0' stimuli. It even appears to be too powerful, having wasted a large number of levels that are not necessary.

However, in the case of graph 'c', the peaks are not so readily separable, and while reasonable boundaries can be drawn, these show a possible need for a machine with more potential levels in each cell (or alternatively more training.)

Summary

It can be seen from the above experiments that several quantitive measures have been developed :

1 The suitability of the resolution of a machine for differentiating between the peaks of consistent, inconsistent and insufficient training,

2 The quality of the training itself in the two aspects :
- sufficiency
- consistency

3 The quantity of the training with regards to the possibility of excessive training in two aspects :
- excessive, but good training (the machine is over powerful, and can easily resolve the training algorithm accurately),
- excessive, but poor training (confusing the machine, which is incapable of resolving the algorithm).

It is believed that such techniques as those developed here give considerable insight into the potential power of LPP machines.

6.5 A Summary of the Experimental Work

This chapter, in conjunction with Chapters 3 and 5, completes the documentation of the experimental work performed on the various LPP machines. The problems of performing such experimental work can be summarized in three major considerations :

1 The basic machine has an unlimited number of variations possible in terms of its internal structure, operation and size,

2 The range of tasks the machine can perform is potentially large, and also covers many different types of operation,

3 The results cannot be easily quantified for interpretive purposes. This is because the inputs and outputs (as in all digital picture processing) are in the form of spatial arrays of pixels, which do not lend themselves readily to assessment by rigorous means.

The first of these two points when combined together give a multitude of combinations to be explored - each new combination requiring evaluation by a method that is frustrated by the third point above. However, in spite of these difficulties, experiments can be performed that do lead to conclusive results, and predictions from

extrapolations have been fulfilled. This has confirmed the validity of the predominantly pragmatic analysis used throughout these experiments.

The Experiments 1 to 17

(Note that Experiments 1 to 12 were all devoted to parallel testing.)

The initial proving run of Experiment 1 showed that the concept of a trainable picture processing system was possible, and that it could work satisfactorily after training by examples alone.

The immediate realisation that the performance was greatly dependent on the training received led to Experiment 2, where the quantity of training was altered, and the effect on the results noted.

The initial internal state of the machine also had a considerable effect on the final results, as shown in Experiment 3.

A variation in the memory matrix cell structure and operation (the 'heart' of the machine) was attempted in Experiment 4.

Experiment 5 showed how the operation also depends on other internal parameters, such as thresholds for comparison; and also illustrated grey-scale processing.

The range of tasks was extended to illustrate the breadth and generality of picture processing applications that could be attempted with these machines in Experiment 6.

A new system for processing the memory matrix cell contents was adopted in Experiment 7 that appeared to be near optimal in terms of memory space requirements,

processing time and computational effort for this particular machine.

Rotation and reflection of the input window in training were introduced in Experiment 8 as a further enhancement to extract more information from the training set.

The effect of the window scan direction and the particular task attempted - where relevant - was illustrated in Experiment 9.

Experiment 10 used a smaller (5-bit) window as a method of introducing the technique of examining the memory matrix directly. This was used as an aid to interpretation of the machine's behaviour thereafter.

Once a more rigorous method of evaluating the training quality became available, Experiment 11 investigated the correspondence between the training set size, performance and the 'figures of merit' derived from this technique.

The use of 'down-loaded' memory matrices as a means of inserting the required algorithms into the machine was attempted in Experiment 12. However, this did not constitute any form of machine self-training or learning.

Experiment 13 was the first to involve sequential testing. In this sub-divided experiment, the training was specially created to be nearer 'optimal', in that it contained examples of a great number of features processed, and thus defined the required task closely. This experiment also made detailed comparisons of the different modes of operation available : viz. parallel and sequential processing, and the case of feedback.

A different type of memory matrix addressing format was used in Experiment 14, which was computationally more

costly, but more efficient in memory requirements.

Tasks that do not involve shape analysis were attempted in Experiment 15. These indicated the small window size used becoming restrictive, and that further work would benefit from a greater finesse in digitisation.

Experiments 16 and 17 concentrated on the behaviour of the LPP machine when 'asked to do the impossible'. The machine could detect and measure inconsistent training, and these experiments quantified this effect and its implications.

Experiment and Format Lists

This set of Experiments 1 to 17 is summarized in a list in Fig 6.28, which may be useful in following the development of the practical work contained herein. There is also a list of the major variations in the LPP machine formats that have been used (F1 to F9) in Fig 6.29.

This concludes the practical experimental work.

Preliminary Experiments

Exp.No.	Section	
1	3.4	Proving Run (Initial experiment)
2	3.5	Training Set Size (1 and 8 pairs)
3	3.6	Initialisation ('0' or '1' in M.M)
4	3.8	Two and Many Valued M.M Cells (One threshold)
5	3.9	Variable O/P Threshold and Grey Level Outputs

Main Set of Experiments

6	5.2	A Range of Separate Picture Processing Tasks (Clean, Invert, Thin, Thicken)
7	5.3	Tri-state/Bi-state M.M Cells
8	5.5	Augmenting Training by Window Symmetry Operations
9	5.6	The Effect of Scan Direction on an Anisotropic Picture Processing Task (with and without r+r)
10	5.7	5-bit Window and the Direct Examination of the Memory Matrix (5-bit/9-bit)
11	5.9	The Variation of TP with Training Set Size
12	5.10	Down-Loaded Memory Matrix ('Edge')

Exps. mainly on Training Methods

13	6.1	Training by specially prepared Examples (Parallel/Sequential Processing with Feedback)
14	6.2	Variations in the Addressing Function (k,t,PO)
15	6.3	Locating and Tracking of Objects
16	6.4	LPP Machine 'Neurosis'
17	6.4	Measuring of Training Quality

Fig 6.28 The Experiments 1 to 17

Format No.	Description	Experiments
F1	16^2, 9-bit Window, Bi-state M.M, Set or Clear	1,2,3,4
F2	16^2, 9-bit Window, Multi-levelled M.M, Increment or Decrement	4,5
F3	32^2, 9-bit Window, Bi-state M.M	6,7
F4	32^2, 9-bit Window, Tri-state M.M, No (r+r)	7,8,9,10
F5	32^2, 9-bit Window, Tri-state M.M, With (r+r)	8,9,11,12, 13,15
F6	32^2, 5-bit Window, Tri-state M.M, No (r+r)	10
F7	32^2, 9-bit Window, Bi-state M.M, (k,t,P0) Addressing Function	14
F8	32^2, 9-bit Window, 4-state M.M, 0,?,N,1 to show Inconsistent Training	16
F9	32^2, 9-bit Window, 2x16 state counters in each M.M cell, to show Training Consistency in Detail	17

Fig 6.29 The LPP Machine Formats F1 to F9

CHAPTER 7

THE SIMULATION OF LEARNING PICTURE PROCESSORS ON A MICROCOMPUTER

7.1 The Need for Software Simulation of the LPP Machine

It is clear from the experiments described in the preceding chapters that it has proved possible to construct successfully working LPP machines. There are several alternative methods by which such working machines could have been created. These fall broadly into two categories :

1 the actual construction of the hardware, as specified in the layout and operation definitions of the machine,

2 the simulation of these machines on a general purpose emulator, such that it performs as the equivalent hardware.

In research work at this early stage of a complex system's life, it is generally accepted that the latter alternative offers more benefits as will be briefly discussed below.

The Advantages of Simulation

The predominant factor in judging the cost of the software or hardware approaches to generating a complex machine lies in the flexibility of either system. Where modifications are to be made to the basic machine, and where

in addition the number of such modifications will be large, the solution that facilitates these changes will be preferred. The software simulation approach provides this facility and also enables the 'virtual machine' (the machine being simulated) to be placed at the centre of an existing computing system, with the resultant access to facilities already available. There are several examples of this type of facility :

1 the means to feed picture information into the virtual machine,

2 the storage capability : it will be of great use to be able to store picture (and other) bulk data permanently,

3 the display of visual data in a recognisable form before, during and after the processor has acted upon it.

The speed with which an initial design and subsequent modifications can be made is another factor. The software approach allows generality with only reprogramming. The only apparent disadvantage in simulation at this early stage is the processing speed of the resultant machine. A simulated machine necessarily runs significantly slower than a purpose built hardware device. However, there is usually the opportunity for estimating relatively accurately the speed of an equivalent hardware device, once the simulation is running. Consequently, this need not be a problem in the research laboratory which is not running in a 'real time'

environment.

It should be mentioned that a compromise between hardware and software is possible in simulation, and is often an optimal solution. That is, within a framework of a simulated machine, some sub-sections can be composed of hardware that is interconnected to the simulating computer. An example in this case would be the use of a shift register type processor (61) for the extraction of windows of pixels at high speed. This could be built to the specification of the machine being simulated, and accessed by the rest of the simulating program. This approach is often superior to either a totally hardware or software based machine in terms of overall efficiency.

However, this early work on the LPP machine uses the predominantly software based approach to simulation as described below. The underlying hardware equivalents are only referenced for the purpose of performance anaylsis or for suggesting new versions of LPP machines.

7.2 The Specification of a Suitable Simulator

The Simulation System is organised as one main section and a number of auxiliary sections to facilitate operations. The main section includes the means for setting up, training and testing of a virtual LPP machine, such that it behaves as an actual machine. The auxiliary sections permit the handling of picture data on the various devices available, examination of the internal state of the machine, and the cascading and feedback of pictures around such machines. These sections are expanded upon below.

7.2.1. Train/Test Section

This constitutes the main core of the simulator, and enables the operator to define and exercise a LPP machine in either of its two basic modes of operation : training or testing.

In the case of training, the operator supplies the simulator with the relevant information regarding the source devices for the input and example pictures and the number of such pairs. In testing, the source of the test input pictures and the destination of the resultant output is required.

The exact format of the LPP machine depends predominantly on the version of the simulator currently in operation and, to a lesser extent, on some operator selectable options specified in response to prompts from the simulator. (It will be appreciated that many versions of the simulator were created, as a result of many LPP machine variants being created for experiment. These vary in details of operation, yet the major principles remain the same.)

7.2.2. Data Handler Section

This section handles picture data, to facilitate the setting up and movement of picture files to be used by the above Train/Test section. There are several requirements within this section, namely to move, edit, display or print pictures under operator control.

The movement of pictures occurs from a source device (video camera, disk storage, paper tape reader) to a destination device (disk or paper tape) and is usually used to create or modify files or sequences of pictures in the required place and device. The operator can specify the

source, destination, number and other picture parameters before the transfer occurs.

The editing facility allows pictures to be moved in a manner similar to the above, but pauses with each picture displayed on a VDU for editing by the operator under cursor control. The cursor can be moved over any pixels, which can be altered as desired, the new picture then continuing on to its destination.

The display of pictures involves the movement of pictures from a source to a VDU for visual inspection by the operator. This would generally be the inspection of a picture file before or after processing by the LPP machine. A number of pictures can be displayed simultaneously, dependent on the picture size used.

The printing of hard copy versions of these low resolution (16^2,32^2) pictures can be effected by moving pictures from a source device to a print buffer, and then to a character printer to form a permanent record.

7.2.3. Memory Matrix Handler

This section allows the operator to examine, change, load, store or print the simulated contents of the memory matrix of the virtual LPP machine. This facilitates interpretation of the machine's operation, by examination of the resultant memory matrices after training, and to make selective changes to it to see how this affects performance.

Complete memory matrices can be generated by the operator and tested, hence by-passing the normal 'train-then-test' cycle. This can be a valuable aid to understanding the machine's characteristics.

These memory matrices can be stored on disk and subsequently re-loaded, to enable re-runs of tests without repeating the same training phase on each occasion. Training can also be halted at stages throughout the training period, and the memory matrix at each stage stored, to investigate how the information in the memory matrix develops.

7.2.4. Cascading and Feedback of Data

The use of picture storage media as an intermediate store allows the cascading of several (possibly different) LPP machines, and allows the use of feedback. This may be effected by one of two methods. Each pass through each machine stage may be treated as a separate test run, by temporarily storing the intermediate results, and then running the next LPP stages on this data. Alternatively, the later versions of the simulator have the facility for feedback within the internal working store of the simulator, thus avoiding the use of intermediate storage and the corresonding processing time penalty. In either case the result is the same and illustrates the operation of these multiple pass LPP machines.

This concludes a description of the facilities included in the simulator. The heirarchical structure of these sections and the functions of the sub-sections is shown in Fig 7.1. The exact construction of such a software tool depends on the hardware available, as much of the processing is related to hardware generated (visual) data. Consequently, there now follows a description of the hardware upon which this simulator will run.

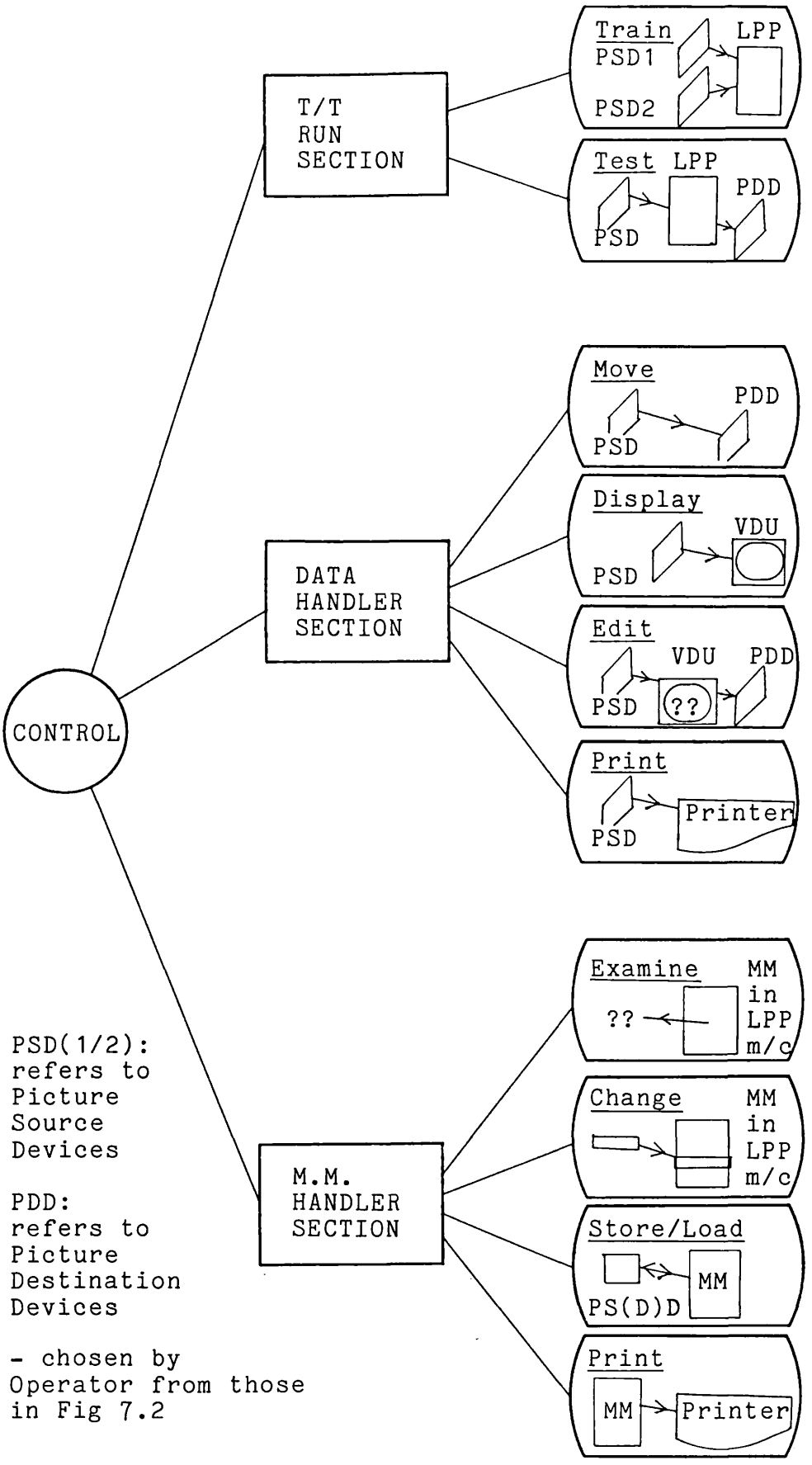


Fig 7.1 Sections of the Computer LPP Simulator

7.3 The Hardware Available for Picture Processing

In order to implement a simulator as described above, there is an obvious need for a suitable hardware environment. The choice of computers was restricted to one of two general categories :

1 large, remote mainframes, accessed via modem links,
2 small local machines, accessed by any local devices.

The former have the advantages of great processing speed and power, the availability of large quantities of backup storage and high system integrity and reliability. While these features are all highly desirable, they are offset by the single major disadvantage of such systems - the speed and methods available for transmitting large quantities of data to and from the system. In the field of visual picture processing, it will be appreciated that vast quantities of data are processed, necessitating high speed data transfer if experiments are to be performed in a realistic time.

Consequently, the availability of high speed video input and output devices connected to a small local machine resulted in the choice of such a machine for these experiments. These high speed devices were of the form of special purpose interfaces to digitize pictures from a conventional video camera, and to display digitized pictures. Similar systems have been described already in the literature (14).

The computer system used was a conventionally structured machine, organised around a single Motorola M6800 microprocessor (52), incorporated into a MSI-6800 micro-

computer (49,48). The main elements of the relevant hardware are described below :

1 CPU with 1 MHz clock
2 48K bytes RAM, 1K byte ROM
3 2K memory mapped RAM, displayed as 64x32 video alphanumeric characters
4 Video input interface from camera
5 Floppy disk storage system
6 Paper tape punch
7 Paper tape reader
8 Control terminal
9 Printer

These devices are interconnected in a conventional bus structure as shown in Fig 7.2. This hardware forms the system on which the simulator described is implemented.

7.4 The Structure and Operation of the Simulator

A simulator with the properties described in Section 7.2 was set up to run on the hardware described in Section 7.3. In addition to these facilities, the additional software tool required was a self-assembler package. This takes the form of a program, resident in the computer which enables the creation of an assembly code file (the source listing of the simulator) and the conversion of this file to a machine code file executable by the computer. The use of assembly code (and hence machine code) as opposed to a high level language interpreter or compiler on this machine was to maximize the available size, power and speed of

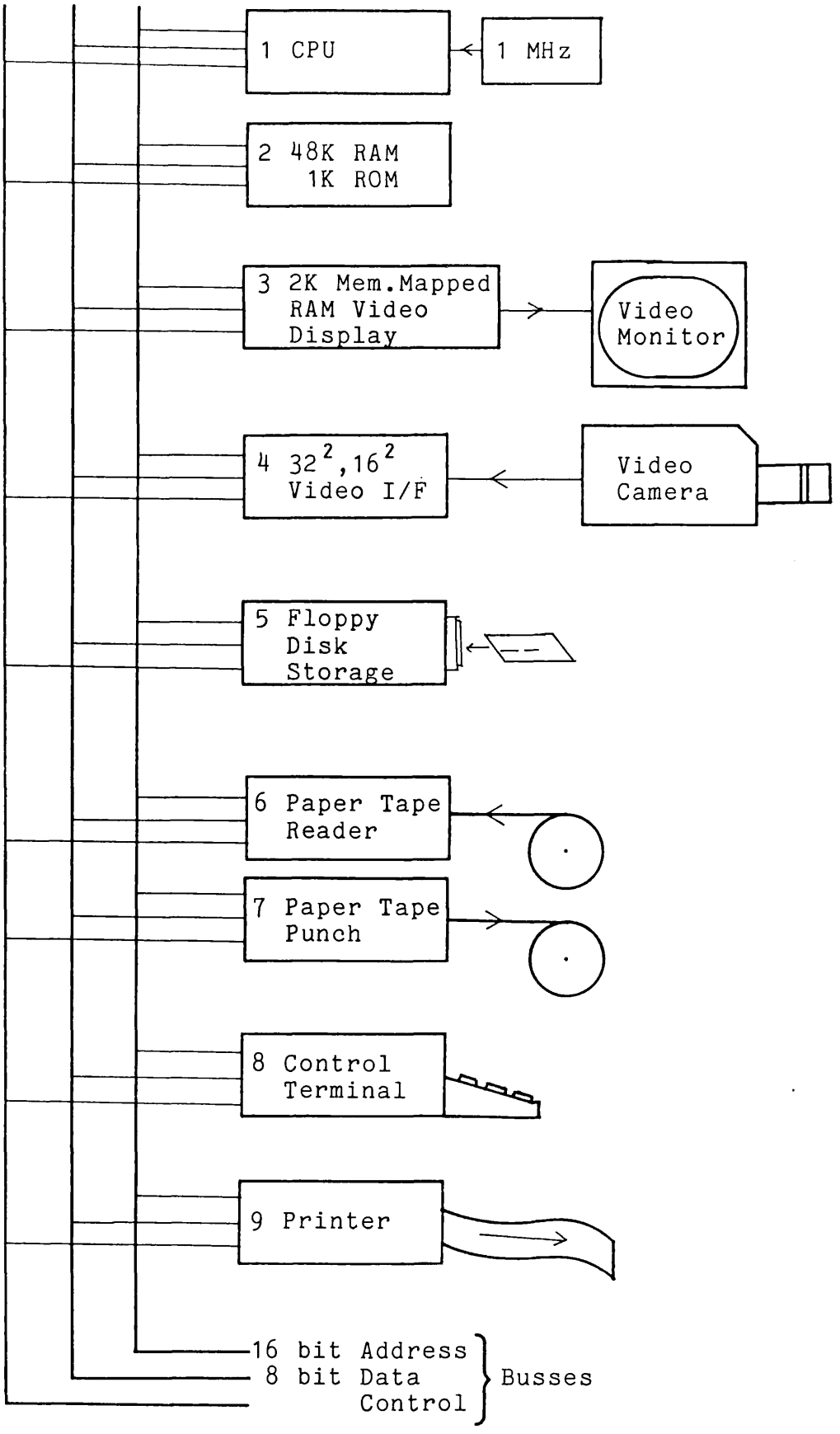


Fig 7.2 Hardware Used for Picture Processing

processing. Using this assembler, the source code for the LPP simulator was created, debugged, modified and tested.

General Simulator Layout

The simulator is divided into three main parts, predominantly to facilitate the handling of a large program such as this, which comprises some 2000 lines of assembly code. These parts may be defined in terms of the modules in Fig 7.1 :

Part 1	'LPPF5'*	Control Section
		Train/Test Run Section
		Data Handler

(* That is, this version simulates the F5 LPP m/c)

Part 2	'LPPMM1'	Memory Matrix Handler
Part 3	'LPPSR21'	A set of Sub-routines for :
		I/O Routines
		Calculations
		Display Routines
		Internal LPP Functions

A complete annotated assembly listing of these three parts of the simulator appears in Appendices 1a-c. Assembled code from each part is loaded into the machine at run time and these divisions become transparent. The remaining structure that is visible from the operator's viewpoint is decribed below. (In what follows, the labelling of each module follows the mnemonic name given in the listing in

Appendix 1. These names are usually self explanatory.)

Operation

The program takes the form of an interrogating interpreter. Once entered, actions are carried out in accordance with the operator's responses to the simulator's queries. The sequence of events that causes the simulator to enter the various modules generally takes the following form :

1 The simulator announces the module in which it is currently waiting (the 'current' module)

2 The simulator provides a list of the modules that can be entered from the current module and requests that a choice is made ('menu-driven')

3 the operator responds with a valid reply, defining which new module is to be entered. (Invalid responses are ignored and control returns to '1' above.)

4 The desired module is entered

5 Dependent on the module, the simulator requests all the relevant parameters for the operation to proceed
(For example, if in the Data Mover Section which transfers pictures between storage devices, the simulator would request the source and number of pictures to be moved and their destination)

6 Once all the relevant parameters have been satisfactorily supplied, the simulator executes the

function, and returns to the control level '1' above.

The actual modules, their functions and options available are listed below. This should be studied in conjunction with Fig 7.1 above.

Control and Three Main Modules

N.B. 1. The names used for these Sections in the listings in Appendix 1 are shown thus : 'NAME',

2. 'Data' below refers to Picture Data, unless 'Cell Data' is specified, referring to Memory Matrix Cell contents.

'LOOP'	Train/Test Run Module
can enter :	Data Handler Module
(Control Section)	Memory Matrix Handler Module
'TTRUN'	Train Section
can enter :	Test Section
	Control Section
'DHAND'	Move Data Section
can enter :	Display Data Section
	Edit Data Section
	Print Data Section
	Control Section
'MMHAND'	Print Addresses Section
can enter :	Print Cell Data Section
	Keyin Cell Data Section
	Totalize Cell Data Section
	Control Section

Train/Test Run Module

'TRAIN' requests :	Source and number of IPP Training Patterns Source of EXP Training Patterns Value of Picture Field Edge Point to use Preset Value of M.M (if applicable)
then :	Trains Virtual Machine with Data as Specified Requests if and where resultant M.M is to be stored on disk Returns to 'TTRUN' Module

'TEST' requests :	Source and Number of IPP Test Patterns Destination of OPP Patterns Serial or Parallel Operation Pause or not after each Test Pattern (for visual Inspection) Value of Picture Field Edge Point to use If and from where M.M on disk is to be loaded into Virtual machine Number of Feedback passes to be made for each pattern
then :	Tests Virtual Machine with Data as specified Returns to 'TTRUN' Module

Data Handler Module

'MOVED' requests :	Source and number of patterns to be moved Destination of Patterns
then :	Moves Data as specified Returns to 'DHAND' Module
'DISP' requests :	Source and number of Patterns to be Displayed on L.H. Side of VDU Source of Patterns to be Displayed on R.H. Side of VDU
then :	Displays Data as Specified Pausing between each Pair Returns to 'DHAND' Module
'EDIT' requests :	Source and Number of Patterns to be Edited Destination of Resultant Patterns
then :	Edits Patterns singly on VDU and Moves Results to Specified Destination Returns to 'DHAND' Module
'PRINTD' requests :	Source and Number of Patterns to be Printed on L.H. Side of Paper Source of Patterns to be Printed on R.H. Side of Paper
then :	Prints Patterns as specified Returns to 'DHAND' Module

7.5 The Introduction of LPP Machine Variants into the Basic Simulator

The simulator described above has been structured as a set of nested subroutines, called as required. A specific, fixed set of subroutines constitutes a particular LPP simulator. In changing the LPP machine being simulated, the change necessary in the simulator is facilitated by this nested structure. An example of this is described below, although it will be appreciated that many more versions of the simulator were created than are fully documented here.

At the lowest level of subroutine calls, the functions used in the Train and Test Sections follow the internal functions described in general terms in Section 4.2. For example, the subroutine labelled 'CADDR' (line 5000 in the listing of 'LPPF5' in Appendix 1a) is exactly equivalent to the function 'f_2' in Section 4.2, where the window contents (the variables 'P0' to 'P8' in 'LPPSR21' lines 3050 to 3130) are re-arranged to form the memory matrix address ('ADDR' in 'LPPSR21' line 3180). Generally, each of these functions that describes the internal working of a particular variant of the LPP machine constitutes a single subroutine.

Consequently, it only requires a change in a particular well-defined subroutine (or sometimes a new subroutine to completely replace the old one) to effect a change in the LPP machine specification. In the example above, a change is made to the routine 'CADDR', which is then called as and when required in the train and test cycles of the simulator. Assuming it still takes the correct input variables (P0-P8) and produces the output variable (ADDR) in the correct format and position, no further changes need be made to the

remainder of the simulator to emulate the new machine. An example of such a change by the insertion of a new subroutine was described in Section 6.2 Experiment 14, where the LPP machine format was changed from F5 to F7 by this method.

(The generation of a wide variety of machines becomes simple using this method, and further confirms the greater efficiency of simulation as a solution to this problem.)

7.6 An Example of the Simulator in Operation

To facilitate an understanding of how the simulator works in practice, an annotated example of the interactive exchange that occurs between the simulator and the operator is presented. An actual experiment from this thesis will be performed : Experiment 11 (Part 2) - the training with four pairs of patterns and subsequent testing with one pattern (see Section 5.9). An examination of the resultant memory matrix is then made, using the 'Totalize' facility. (This counts the number of cells set to each of the three - in this case - possible values.)

The printout below takes the following form :

1 the outputs generated by the simulator are in upper case letters, not underlined,

2 the operator's responses and commands are in upper case letters, and underlined,

3 the annotations explaining the actions are in lower case letters in parentheses and indented,

4 the current memory mapped display (showing the patterns being processed) is shown where relevant to the current operation. The pattern format used here is 32^2, two such patterns can be displayed simultaneously on the VDU.

Sample Printout - Experiment 11 (Part 2)

(All numbers are in hexadecimal)

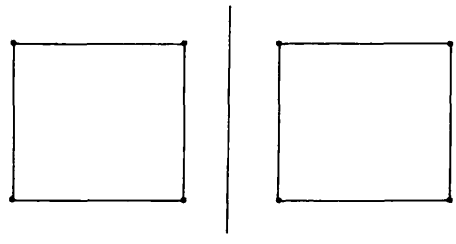
```
*

*G 0100
LPP SYSTEM 10 - MEMORY MATRIX FORMAT F5
```

(system entered, version of simulator and format announced)

```
VDU TO BE CLEARED ("Y" OR "N") ? Y
```

(request to clear memory mapped VDU, response : Y(es), hence appears as -)



```
CONTROL LOOP - DATA HANDLER , T/T RUN , M.M. HANDLER ? T
```

(control loop prompt, command to enter t/trun section)

```
T/T RUN SECTION -
TRAIN OR TEST RUN ("N" OR "T") ? N
```

(comand to enter train section)

```
INPUT PATTERN (IPP) - SOURCE OF DATA -
TAPE , DISK , CAMERA , X(DUMMY) ? D
STARTING TRACK AND SECTOR (TTOS) ? 0000
NO. OF SOURCE CHARACTERS (HH) ? 04
INCREMENTAL STEP IN SOURCE FILE (HH) ? 01
```

(relevant parameters requested for ipp source file : response is that characters will come from disk, track/sector address 0000, four patterns separated by a step of '1' i.e consecutively placed on disk)

EXAMPLE PATTERN (EXP) - SOURCE OF DATA -
TAPE , DISK , CAMERA , X(DUMMY) ? D
STARTING TRACK AND SECTOR (TTOS) ? 0604

INCREMENTAL STEP IN SOURCE FILE (HH) ? 01

(as above for exp source file :
however it is assumed that there is the same
number of patterns as ipp as they come in pairs)

EDGE POINTS ("1" OR "0") ? 0

(request for picture field edge point value)

PRESET MEMORY MATRIX ("Y" OR "N") ? Y

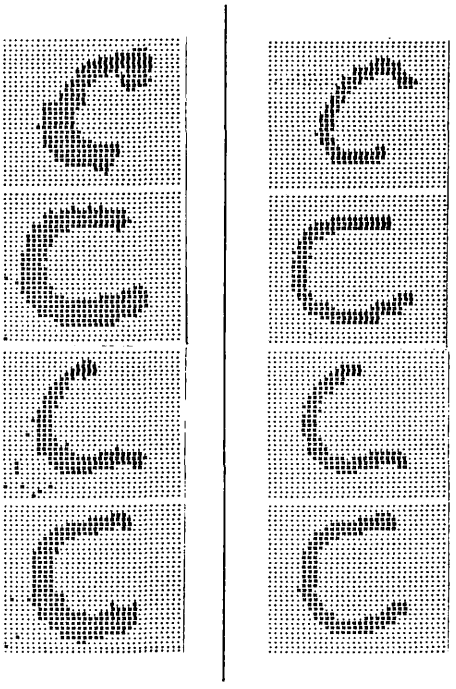
(preset m.m cells to 'bit x' value)

READY (X) ? Y

(pause prompt, to ensure all relevant devices have
been correctly initialised eg. disks inserted.
It is here, now all the relevant information has
been supplied, that the actual training of the
virtual machine occurs. VDU displays ipp+exp pairs
as they are processed -)

IPPs

(four pairs
of 'raw'
and 'clean'
letters
'C')



EXPs

STORE MEMORY MATRIX ("Y" OR "N") ? Y
STARTING TRACK AND SECTOR (TTOS) ? 4000
READY (X) ? Y

(after training, option is given to store m.m
contents on disk for later examination or
reloading. This is done at t/s 4000 - a scratch
area on the disk)

CONTROL LOOP - DATA HANDLER , T/T RUN , M.M. HANDLER ? T

(return to control loop - re-enter t/trun section)

T/T RUN SECTION -
TRAIN OR TEST RUN ("N" OR "T") ? T

(enter test section)

INPUT PATTERN (IPP) - SOURCE OF DATA -
TAPE , DISK , CAMERA , X(DUMMY) ? D
STARTING TRACK AND SECTOR (TTOS) ? 0703
NO. OF SOURCE CHARACTERS (HH) ? 01

(ipp source file details - one pattern from disk at t/s 0703)

OUTPUT PATTERN (OPP) - DESTINATION OF DATA -
DISK , X(DUMMY) ? D
STARTING TRACK AND SECTOR (TTOS) ? 4100

(opp destination details - results will be placed on disk at t/s 4100 - scratch area - for later examination, comparison, printing or use as input to further machines)

SERIAL OR PARALLEL ("S" OR "P") ? P

(testing mode option)

PAUSE AFTER EACH PATTERN ("Y" OR "N") ? N

(facility to pause if desired between test patterns to facilitate visual inspection while a test set runs through machine)

EDGE POINTS ("1" OR "0") ? 0

LOAD MEMORY MATRIX ("Y" OR "N") ? N

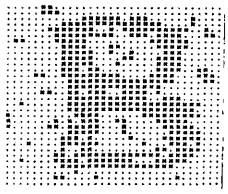
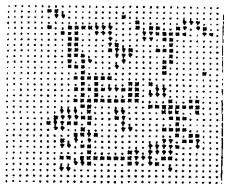
(facility to load m.m from disk, used if machine had not just been trained above)

NO. OF PASSES (HH) ? 01

(number of feedback passes to be made by each test pattern)

READY (X) ? Y

(here, the virtual machine is tested now all the required information has been gathered. The VDU displays the test patterns (ipp+opp together) as the machine proceeds through the test set, originally shown in Fig 5.28)

IPP   OPP

CONTROL LOOP - DATA HANDLER , T/T RUN , M.M. HANDLER ? M

(return to control loop, and enter
m.m handler section)

MEMORY MATRIX HANDLER -
PRINT ADDRESSES , DATA SETS , KEYIN M.M. , TOTALIZE
OR CONTROL ? T

(function required ? - 'totalize')

LOAD MEMORY MATRIX ("Y" OR "N") ? N

(option to load m.m from disk, but is still
in core due to above training, hence no need
to re-load)

NO. OF M.M. LOCS SET TO BIT 0 - 09A
NO. OF M.M. LOCS SET TO BIT X - 14A
NO. OF M.M. LOCS SET TO BIT 1 - 01C

(simulator examines m.m in core, counts
number of cells set to 0, X and 1, and ouputs
results on terminal in hexadecimal)

MEMORY MATRIX HANDLER -
PRINT ADDRESSES , DATA SETS , KEYIN M.M. , TOTALIZE
OR CONTROL ? C

(return to control loop)

CONTROL LOOP - DATA HANDLER , T/T RUN , M.M. HANDLER ? _

(simulator awaiting further commands)

This concludes this example of the simulator listed in Appendix 1 in operation. As stated above, many versions of the simulator were created to simulate the many different versions of the LPP machine used. However, all of these simulators operated in a manner similar to that shown above.

CHAPTER 8

CONCLUSION

8.1 Introduction

This chapter is divided into several parts. The first will detail further experiments that would have been attempted immediately following the experiments already contained herein. A broader range of experiments and variations will also be suggested, if these investigations were to be developed to any greater extent.

The work contained in this thesis and the results which spring from it are then summarised. The implications of these results are considered in relation to the future of trainable systems in this and other applications in particular and artificial intelligence in general.

8.2 Detailed Further Work

Throughout these experiments, there have been references to LPP machine developments that might be usefully investigated. Chapter 4 represents an attempt at formalizing a range of possible alternatives regarding the detailed structure and operation of these machines. There, the suggestion was made for cascaded machines, possibly with feedback around multiple stages - the 'compound' machine (Section 4.6). This promises to exhibit interesting behaviour, even if extremely complex to interpret. However, this would form a valuable attempt at synthesizing a

learning machine composed of so many layers that its behaviour could not be easily deduced before it operated. Such a machine represents a 'deep' machine, with many learning layers between input and output, as opposed to 'broad' machines (with fewer layers, yet wide data paths between inputs and outputs) which have been shown to exhibit coherent 'intelligent' behaviour. These successes may lend inspiration to attempt these deep, compound LPP machines.

Other alternatives that would have been attempted involve the use of different (predominantly larger) window sizes, following from the fact that the final experiments illustrated the small window size to be a limitation. Variations in the pixel tessellation (eg. a hexagonal lattice) could also have been attempted, yet no immediate benefit or loss was envisaged from this alternative.

Changes in the experimentation were suggested by the arrival of a new video interface, unfortunately too late for inclusion in this experimental work. This is a new system for digitizing video signals to a higher resolution of 128^2 8-bit pixels. These 8-bit pixels are displayable as one of 256 grey levels, and an example of an image produced on this system is used in the frontispiece of this thesis. Grey scale processing should be possible with the LPP system. This assumes that the address generated from the window of grey scale pixels is kept to a manageable size, to avoid the need for an enormous (and thus untrainable) memory matrix. Careful choice of the size of window, number of grey levels and the transformation from window to address, should enable

a practical trainable grey-scale picture processor. (The new system also has pseudo-colour outputs, which could be usefully employed in displaying processing by examples.)

The tasks that could be attempted include grey scale stretching, colour falsification or modification. These would be trivially simple to implement as a window of one pixel could be used - generating a memory matrix address directly from each pixel. More complex processing, taking a window of multi-valued pixels could implement correspondingly more complex tasks. Any task that has a local spatial dependence could be processed by such a window oriented machine. Examples of this in fields other than picture processing are audio frequency spectrograms (possibly of speech) which are essentially two dimensional images, and consequently potential sources of suitable data. The problem here may be the provision of suitable examples of processed speech.

This may now be stated as a sufficient condition for such machines : if any well-defined task of this type (albeit by an unknown method) is capable of being illustrated by examples, then this task may be implemented using these trainable systems.

An alternative development for further work is the construction of working hardware, rather than a simulation. The realisation of near instantaneous processing speeds (for small pictures), pipelined (cascaded) processors and possibly larger frame storage are exciting possibilities here.

8.3 Broader Experimental Work

In Chapter 4 it was stated that modifications to the 'General LPP Machine' could be envisaged that were beyond the realms of the 'generalized' description given therein. The example was given of a window which scanned in a non-orderly fashion, but rather followed some feature of the picture content. This example is worthy of investigation, as it represents a potentially very powerful processing system which concentrates its action in specific local areas. It is often the case in picture processing, that if a local function is to be executed, local examination of the picture is not only necessary but more efficient. However, this will reduce the machine's generality, and so this cost must be examined closely.

It must be remembered that radical changes in the machine, whilst improving the performance in a particular application, may make it 'specialized'. The attraction of the general LPP machine as described in Chapter 4 lies in its task independence, by virtue of its internal arrangement. Consequently, many of the behavioural results gained by experimentation on a particular task are truly general, and thus applicable to the machine when acting on any task.

Many new techniques of image description by machine are emerging, often to increase efficiency in data storage. An example of this is the 'quadtree' or 'pyramidal' representation (76) where the resolution of samples is variable over the image, dependent on the image content. This method of data reduction could possibly serve as an alternative to the use of a scanning window in the

overriding requirement of such systems : the need to greatly reduce the quantity of data coming in. However, a careful choice of organisaion of the data rearrangement must be made to ensure that the resultant machine remains :

- trainable by example,
- able to generalize, and
- capable of actually producing coherent outputs.

In fact, these requirements which have so far been applied to a spatial window could be extended to a system acting on data extracted from a temporal window. This may have possible applications in stochastic systems, or communications, where a stream of data in time is to be processed. Again, the problem will lie in providing suitable examples to train the machine.

8.4 A Summary of the Experimental Work

The first chapter introduced computers applied to various tasks requiring intelligence. An example of this is the field of picture processing, where much research work has already been done in the last twenty-five years. The many alternative approaches and methods were discussed, two of which are synthesized here into a new type of machine - the trainable picture processor. These techniques are :

1 picture processing by look-up tables

2 RAMs used as learning machines

The second chapter suggested the layouts needed to implement LPP (Learning Picture Processor) machines as a practical reality. Some initial experimental investigations showed this to be a practicable method of using such machines. Variations in the internal and external conditions of operation resulted in different performances being observed and discussed.

A general system for developing and specifying a broad range of variations on this theme was proposed. This system was then used to attempt more ambitious investigations in the fifth and sixth chapters. These gave insight into the more subtle behavioural aspects of the machine. Techniques were also developed (other than merely 'observing output') to investigate the internal states and resultant behaviour of these machines. Some general conclusions and results were drawn from this investigation, summarized separately in the section below.

A description was given of the hardware and software system used to simulate the LPP machines throughout these experiments. The advantages and reasons for simulation were also discussed, and an example of the operation of this simulator was given.

Finally, this chapter suggests further work that might be attempted on these systems. Some of the wider applications are mentioned, in which such learning processors could also be applied.

8.5 A Summary of the Main Results

The main results gathered from these investigations are summarized below :

1 It is feasible to create a learning machine that will process pictures, having been trained only by experience,

2 The process performed will reflect the process illustrated by the examples provided,

3 A single machine can perform a range of tasks by re-training for each task alone,

4 Different complexities of machines can be envisaged. To a certain extent, the more sophisticated the machine the better the test performance, although diminishing returns may eventually become evident,

5 Such machines are capable of generalisation - the ability to process previously unseen pictures,

6 The internal size of a sequentially scanning machine is independent of the picture size processed, but the processing time is dependent on this size,

7 The performance depends to some extent on the internal arrangements and operation of the machine,

8 The perfomance depends to a much larger extent on the quality and quantity of examples presented to the machine in representing the desired process,

9 Several measurements can be taken from a given training set : quantity, consistency, etc.,

10 Symmetry operations can considerably increase the effective information extracted from the training set,

11 Sequential scanning of inputs will bias the algorithm generation towards the end of the scan period,

12 Much useful information can be gained from direct observation of the machine's memory matrix after training: both with regard to the particular training set and the machine's reaction to it,

13 If severe, but justifiable restrictions are placed on the processing possible, it is feasible to attempt a subset of all possible algorithms by their automatic generation,

14 Special training can be created artificially that is near optimal in expressing a required algorithm,

15 These machines exhibit the usual traits in the sequential as opposed to parallel mode of testing,

16 Feedback can be of great use in certain applications eg. thinning to a unit width skeleton,

17 Totally different types of internal transforms can give comparable test results, yet possibly with improved storage efficiency,

18 The machine can detect size comparable with the window size used,

19 The machine can determine the resolution needed to extract most of the information available in a given training set,

20 A workable machine capable of performing useful picture processing could be ultimately very small and thus cheap,

21 Such a machine, if purpose-built in hardware, has a potentially high processing rate, in both training and testing.

8.6 Implications of These Results

It is possible to forecast some implications from the above results. As the need grows for machine generated pictures in a larger range of applications, there will be an equivalent need for machines to process these images. The possiblity of small, cheap, fast processors of the type described here must surely be one candidate to help fill this need. Allied to the fact that these machines are adaptable and furthermore re-trainable without re-programming, this makes them a flexible, as well as cost-effective alternative.

By judicious choice of the internal size and organisation, a particular performance capability can also be specified, in a compromise between price and performance. Thus machines can be envisaged that have sufficient powers to execute their task without being over expensive. This must represent an ideal potential solution.

However, the problems of training such machines must not be underestimated. Since the performance of the machine depends largely on this, careful consideration must be given here. For example, can the machine be trained 'locally', and thus re-trained on site? Could 'factory-training' be used, with the corresponding economy yet loss of flexibility? Can training examples be provided in sufficient numbers and quality by an external means? These are the questions that must be answered before the use of such trainable machines in practical picture processing applications becomes effective.

8.7 Wider Applications of Trainable Systems

It has been shown how the Learning Picture Processing system is in principle capable of tackling any task. It is proposed to leave the realm of the experimental work done here, and suggest that any signal processing - whatever the media, or whatever the signal represents - can be implemented on such a machine.

Any signals transducible to an electrical media - and thus able to be processed by an electronic device - should also be candidates for such intelligent processing. Sonar, microwave, radio, infra-red and X-rays are all examples of media capable of carrying such signals with high complexity and redundancy, which are already regarded as 'imaging' processes - usually representing two or three dimensional spatial images. Sound, temperature, pressure and other 'non-imaging' signals should also be capable of enhancement by this method.

Consideration of these alternative media for processing by trainable systems highlights the basic minimum requirements for such a machine. These are necessary environmental conditions for the subsequent development of the internal operations to make such a machine a practical reality.

These conditions have been stated before, and are summarized here :

1 A system exists where an orderly transformation in some signal is required,

2 External to the machine, there exists the facility for providing 'examples' of the transformation that is to occur (possibly by provision of 'before and after' examples of signals that have been processed),

3 A training period is allowed prior to processing, wherein the machine is given access to these examples.

(In addition, these signals must all be in a form that can be presented to and produced by the machine.)

If all these conditions are met, then a machine should be able to 'learn by example' to process correctly. It will thereafter be limited by internal consideration, such as - memory size, processing power available (and hence throughput) and the ability to react to the training correctly.

It is interesting to note that not only do these conditions apply to human learning, but also to 'unintelligent' learning. The robot paint sprayer, for example, which has its hand guided by a trained operator is

acting under the above conditions. However, it cannot 'generalize' - deal with later inputs substantially different from those used in training. The above conditions are necessary, but not sufficient for a learning machine. The intelligent machine must derive from its training some higher concept of what is required of it; to enable it to handle gross deviations in testing. The ability to generalize is a well known requirement of intelligent systems.

The experimental work has examined such learning machines in a very limited range of applications. However, it has shed some light on how such a class of machines may be constructed so as to act in an environment defined by the conditions above.

The internal actions that are necessary may be summarized in general terms as :

1 The machine must extract from its training stimuli a representation of what is required of it ('learn'),

2 This must be stored internally in such a manner as to be retrievable in a coherent form for later use ('remember'),

3 The description stored earlier of this operation must be capable of transforming later stimuli in a manner similar to that originally presented to the machine ('perform').

The limitations on the machine in isolation are readily predictable, often dependent on the internal arrangements used. The limitations on the performance of the entire

system (ie. the machine and its environment supplying the signals and examples) depends closely on this ability to provide suitable and accurate examples.

This thesis does not suggest where or how these examples can be produced in general. This is ultimately where the problem in developing this kind of system will lie. The band of applications where such examples can be provided is possibly narrower than might be hoped. However, it is a result of this thesis that if examples can be provided, by whatever means, learning machines can be built to respond coherently and take correct advantage of these stimuli.

8.8 Future Trends in Artificial Intelligence

The machines described herein illustrate some of the possible trends developing in electronic data processing. That is, large numbers of small machines are being used, rather than the reverse. The low cost, great numbers and wide availability of such machines is bound to lead to a situation where cheap, local processing prevails.

The problem will lie not in the hardware, but in the software effort required to drive it all. This is where the techniques of artificial intelligence may be able to help. Much of the burden can be lifted from software engineering if adaptive software and systems can be produced that can 'program themselves'.

This is the goal, not only at the grand levels of artificial intelligence, but also at the lower, yet equally important and useful level of data processing of the type

described here. In fact, it is suspected that there will be many more applications using these small, relatively simple systems with adaptive processing capabilities, than the large all-powerful intelligent machine.

It is also likely that such machines may become ever more divorced from classical 'Von Neumann' computer achitectures. The processors described here, for example, whilst using conventional sub-systems as components (RAMs, registers, gates, etc.) are not organised in the conventional 'stored program' manner. This has already been discussed in the literature (3) as a trend developing in existing learning machines. Undoubtedly, this will develop to a stage where radically different architectures of data processing devices will co-exist, as more knowledge is gained regarding intelligent processes.

The ultimate processor at present - the human brain - still represents this challenge. It is organised totally unlike conventional computers of today, which is a portent of their ultimate and inevitable obsolescence. The present results from scratching the surface at the workings of intelligent machines are however encouraging. Totally different types of computer are on the way. This provides inspiration in the path forward for such work in the future.

ACKOWLEDEGEMENTS

I would like to express my gratitude to my supervisor Dr. E. Roy Davies for his inspiration and guidance which has made this work possible. I would also like to thank the other staff of the Physics Department, Royal Holloway College for their support and provision of facilities for this project, in particular Mr. Brian Tait for his photography.

Finally, I am also indebted to the Science and Engineering Research Council for the provision of a Research Studentship Award to fund this work.

REFERENCES

1 Adams J. and Wallis R.
New concepts in display technology.
Computer August 1977 pp 61-69

2 Aleksander I.
Design of universal logic circuits.
Electronics Letters Vol 2 No 8 pp 319-321

3 Aleksander I.
Digital processing for future industrial automation.
Digital Systems for Industrial Automation
Vol 1 No 1 Summer 1980

4 Aleksander I.
'Microcircuit Learning Computers'
Pub: Mills and Boon Ltd. London 1971

5 Aoki M.
Rectangular region coding for image data compression.
Pattern Recognition Vol 11 No 5/6 1979 pp 297-312

6 Arcelli C., Cordella L. and Levialdi S.
Parallel thinning of binary pictures.
Electronics Letters Vol 11 No 7 April 1975 pp 148-149

7 Batchelor B.G.
'Pattern recognition - ideas in practice.'
Pub: Plenum Press New York USA 1978

8 Batchelor B.G.
Laboratory seeds are sown in industry.
Laboratory Equipment Digest Jan 1980 p 71

9 Batchelor B.G., Brumfitt P.J. and Smith B.D.V.
Command language for interactive image analysis.
Proc.IEE Vol 127 Part E No 5 Sept 1980 pp 203-218

10 Beun M.
A flexible method for automatic reading of handwritten numerals - Parts I and II.
Philips Technical Review Vol 33 1975
Part I - No 4 pp 89-101
Part II - No 5 pp 130-137

11 Bledsoe W.W.
Further results on the n-tuple pattern recognition method.
IRE Trans. Elec. Computers EC-10 Mar 1961 p 96

12 Bledsoe W.W. and Bisson C.L.
Improved memory matrices for the n-tuple pattern recognition method.
IRE Trans. Elec. Computers EC-11 Jun 1962 pp 414-415

13 Bledsoe W.W. and Browning I.
Pattern recognition and reading by machine.
Proc. EJCC 1959 pp 225-232

14 Cady F.M. and Hodgson R.M.
Microprocessor based interactive image-processing system.
Proc.IEE Vol 127 Part E No 5 Sept 1980 pp 197-202

15 Castleman K.R.
'Digital Image Processing'
Pub: Prentice-Hall Inc. New Jersey 1979

16 Chalmers J. and Stein A.
Iterative array for the reprocessing of hand-printed characters.
Electronics Letters Vol 5 No 18 Sept 1969 pp 431-433

17 Chang T.L.
Texture analysis of digitized fingerprints for singularity detection.
Proc. 5th Int. Conf. on Pattern Recognition
Dec 1980 Florida USA pp 478-480

18 Chen C.J. and Shi Q.Y.
Shape features for cancer cell recognition.
Proc. 5th Int. Conf. on Pattern Recognition
Dec 1980 Florida USA pp 579-581

19 Cover T.M. and Hart P.E.
Nearest neighbour pattern classification.
IEEE Trans. Information Theory IT-13 1967 pp 21-27

20 Cordella L., Duff M.J.B. and Levialdi S.
Comparing sequential and parallel processing of pictures.
Proc. 3rd Int. Joint Conf. on Pattern Recognition
Nov 1976 California pp 703-707

21 Davies E.R.
Some basic experiments with multi-layer RAM nets.
Technical Report No PRG/D/41 Oct 1978
Dept. of Physics Royal Holloway College London

22 Davies E.R. and Plummer A.P.N.
Thinning algorithms: a critique and a new methodology.
Pattern Recognition Vol 14 Nos 1-6 (Sp.Iss: 1980 Conf. on Pattern Recognition) 1981 pp 53-63

23 Deutsch E.S.
Thinning algorithms on rectangular, hexagonal and triangular arrays.
Comms. ACM Vol 15 No 9 1972 pp 827-837

24 Dixon R.N. and Taylor C.J.
Automated asbestos fibre counting.
Proc. IoP Conf. on Machine-aided Image Analysis
Institute of Physics Bristol and London pp 178-185 1979

25 Duda R.O. and Hart P.E.
'Pattern classification and scene analysis'
Pub: Wiley USA 1973

26 Duff M.J.B.
Clip 4: a large scale integrated circuit array parallel processor.
Proc. 3rd Int. Joint Conf. on Pattern Recognition
Nov 1976 California pp 728-733

27 Dyer C.R., Rosenfeld A. and Samet H.
Region representation: boundary codes from quadtrees.
Comms. ACM Vol 23 No 3 Mar 1980 pp 171-179

28 Fountain T.J. and Goetcherian V.
Clip 4 parallel processing system.
Proc. IEE Vol 127 Part E No 5 Sept 1980 pp 219-224

29 Garvey T.D. and Fischler M.A.
The integration of multi-sensor data for threat assessment.
Proc. 5th Int. Conf. on Pattern Recognition
Dec 1980 Florida USA pp 343-347

30 Goetcherian V.
From binary to grey tone image processing using fuzzy logic concepts.
Pattern Recognition Vol 12 No 1 1980 pp 7-15

31 Granovskaya R.M. and Bereznaya I.J.
Experiments on human pattern recognition: a hierarchical sign-system approach.
Pattern Recognition Vol 12 No 1 1980 pp 17-26

32 Groh G.
The challenge of picture processing.
Philips Technical Review Vol 38 No 11/12 1978/9
pp 291-297

33 Guentri D. and Norton-Wayne L.
Automatic guidance of vehicle using visual data
Proc. 5th Int. Conf on Pattern Recognition
Dec 1980 Florida USA pp 146-149

34 Haig D.C, Kulick J.H, Challis T.W. and Haig C.M.
Automated screening of chest X-rays: image segmentation and cardiac silhouette analysis.
Proc. 4th Int. Joint Conf. on Pattern Recognition
Nov 1978 Kyoto Japan pp 902-906

35 Hale J.A.G. and Saraga P.
Digital image processing. Chapter 7
in: Batchelor B.G. (ed)
'Pattern Recognition - Ideas in Practice'
Pub: Plenum Press New York 1978

36 Heeschen R, Joseph R, DiBianca F. and Cohen G.
Image processing in computer radiography.
IEEE Conf. on Pattern Recognition and Image Processing
Aug 1979 Chicago USA pp 356-362

37 Highleyman W.H.
An analog method for character recognition.
IRE Trans. on Elec. Computers EC-10 Sept 1961
pp 502-512

38 Hilditch C.J.
Linear skeletons from square cupboards.
in 'Machine Intelligence IV' eds:Meltzer B, Michie D.
Pub: Edinburgh Univ. Press Edinburgh 1969 pp 403-420

39 Hirzinger G, Landzettel K. and Snyder W.
Automated TV tracking of moving objects: the DFVLR-tracker and related approaches.
Proc. 5th Int. Conf. on Pattern Recognition
Dec 1980 Florida USA pp 1255-1261

40 Hoyer A. and Schlindwein M.
Digital image enhancement.
Philips Technical Review Vol 38 No 11/12 1978/9
pp 298-309

41 Hunt D.J.
A feature extraction method for the recognition of handprinted characters.
in: 'Machine Perception of Patterns and Pictures'
Proc. of Conf. by IoP, NPL, IEE held at
NPL Teddington Apr 1972
Pub: Institute of Physics London 1972

42 Ito C.
Figure pre-processing device.
US Patent 4 115 760 Sept 1978

43 Jones R.N. and Fairhurst M.C.
Skeletonisation of binary patterns:a heuristic approach.
Electronics Letters Vol 14 No 9 1978 pp 265-266

44 Kusaka T, Haba Y, Kawata Y, Terashita Y, and Ueno S.
Removal of the atmospheric blurring from remotely sensed earth imagery.
Proc. 4th Int. Joint Conf. of Pattern Recognition
Nov 1978 Kyoto Japan pp 931-935

45 LeMay C.A.G.
Automatic standardization of size, position and other variables for pattern recognition.
in: 'Machine Perception of Patterns and Pictures'
(See ref no.41 above)

46 McVey E.S. and VanTol F.
An experimental PCB drilling system automated by PR.
Pattern Recognition Vol 11 No 4 1979 pp 271-276

47 Marko H, Platzer H. and Stroke G.W.
Optical processing.
Proc. 4th Int. Joint Conf. on Pattern Recognition
Nov 1978 Kyoto Japan pp 155-180

48 Midwest Scientific Instruments
FD-8 Floppy disk drive system manual.
Olathe Kansas 1977

49 Midwest Scientific Instruments
MSI-6800 Computer system Manual.
Olathe Kansas 1977

50 Minsky M. and Papert S.
'Perceptrons - An Introduction to Computational Geometry'
Pub: MIT Press Massachusetts 1969

51 Montanari U.
Continuous skeletons from digitized images.
Journal of the ACM Vol 16 No 4 Oct 1969 pp 534-549

52 Motorola Semiconductor Products Inc.
M6800 Microcomputer system design data.
Phoenix Arizona

53 Munson J.H.
Experiments in the recognition of hand-printed text: Part I - Character recognition.
Proc. FJCC Dec 1968 pp 1125-1138

54 Nappey J.A.
Aspects of n-tuple character recognition for a blind reading aid.
PhD Thesis Brunel University Sept 1977
(Post Office characters referred to kindly made available by Prof. I.Aleksander Brunel University)

55 Nappey J.A.
Jansys: a suite of programs for pattern recognition.
Internal Report AC/R/050 1977
Dept of Elec. Engng. and Electronics Brunel University
Uxbridge Middlesex

56 Nedjl I.F, Gose E.E. and Kaplan E.
Computer aided diagnosis from lung ventilation and perfusion scintigrams.
Proc. 4th Int. Joint Conf. on Pattern Recognition
Nov 1978 Kyoto Japan pp 914-918

57 Niemann H.
Classification of characters by man and machine.
Pattern Recognition Vol 9 No 4 1977 pp 173-179

58 Plummer A.P.N.
'Structural Analysis and Classification of Patterns'
Ph.D Thesis Royal Holloway College London University
April 1980

59 Pullen A.P.
Automatic visual inspection of complex industrial components.
Paper read at BPRA meeting at UCL London 22 Feb 1977

60 Rao C.V.K, Prasada B. and Sarma K.R.
An automatic fingerprint classification system.
Proc. 2nd Int. Joint Conf. on Pattern Recognition
August 1974 pp 180-184

61 Riganati J.P. and Vitols V.A.
Binary image minutiae detector.
US Patent 4 083 035 Apr 1978

62 Riganati J.P. and Vitols V.A.
Two-dimensional binary data enhancement system.
US Patent 4 003 024 Jan 1977

63 Rosenfeld A.
Connectivity in digital pictures.
Journ. ACM Vol 17 No 1 Jan 1970 pp 146-160

64 Rosenfeld A.
Recent developments in image and scene analysis.
Inst. Phys. Conf. Ser. No 44 Chapter 1 1979 pp 42-49

65 Rosenfeld A. and Johnston E.
Angle detection on digital curves.
IEEE Trans. Computers C-22 1973 pp 875-878

66 Rosenfeld A. and Kak A.C.
'Digital Picture Processing'
Pub: Academic Press New York 1976

67 Royston R.J.
Applications in nuclear physics.
In 'Machine Perception of Patterns and Pictures'
Pub: IoP and NPL London April 1972

68 Saedtler E.
Automation of reactor monitoring systems by
means of pattern recognition techniques.
Proc. 5th Int. Conf. on Pattern Recognition
Dec 1980 Florida USA pp 116-120

69 Saraga P., Weaver J.A. and Woollons D.J.
Optical character recognition.
Philips Technical Review Vol 28 No 5/6/7 1967
pp 197-203

70 Sherman H.
A quasi-topological method for the recognition of line
patterns.
Information Processing - the Proceedings of a UNESCO
Conference in Paris 1959. Butterworths London 1960

71 Singer J.R.
Electronic analog of the human recognition system.
Journ. of the Optical Soc. of America Vol 51 No 1 1961
pp 61-69

72 Steck G.P.
Stochastic model for the Browning-Bledsoe pattern recognition scheme.
IRE Trans. on Electronic Computers EC-11 Apr 1962 pp 274-282

73 Stefanelli R. and Rosenfeld A.
Some parallel thinning algorithms for digital pictures.
Journal of the ACM Vol 18 No 2 Apr 1971 pp 255-264

74 Su C.
Pre-processing and feature extraction system for character recognition.
US Patent 4 162 482 July 1979

75 Takagi M.
Biomedical image processing and pattern recognition.
Proc. 4th Int. Conf. on Pattern Recognition
Nov 1978 Kyoto Japan pp 146-149

76 Tanimoto S. and Pavlidis T.
A hierachical data structure for picture processing.
Computer Graphics and Image Processing Vol 4 1975 pp 104-119

77 Ullman J.R.
Experiments with the n-tuple method of pattern recognition.
IEEE Trans. on Computers C-18 Dec 1969 pp 1135-1137

78 Ullman J.R.
Video-rate digital image analysis equipment.
Pattern Recognition Vol 14 Nos 1-6 (Sp.Iss: 1980 Conf. on Pattern Recognition) 1981 pp 305-318

79 Unger S.H.
A computer oriented toward spatial problems.
Proc. IRE Vol 46 No 10 Oct 1958 pp 1744-1750

80 Unger S.H.
Pattern detection and recognition.
Proc. IRE Vol 47 No 10 Oct 1959 pp 1737-1752

81 Verhagen C.J.D.M., Duin R.P.W., Groen F.C.A., Joosten J.C. and Verbeek P.W.
Progress report on pattern recognition.
Reports on Progress in Physics Vol 43 1980 pp 785-831

82 Wilson M.J.D.
Sequential cellular automata.
Technical Memo. No N/R/003 Oct 1975
Dept of Elec. Engng. and Electronics Brunel University Uxbridge Middlesex

83 Wilson M.J.D. and Aleksander I.
Pattern-recognition properties of RAM/ROM arrays.
Electronics Letters Vol 13 No 9 Apr 1977 pp 253-254

APPENDICES

Appendix 1 Assembly Listing of LPP Software Simulator

App. 1a	Part 1	'LPPF5'	(8 sheets)
App. 1b	Part 2	'LPPMM1'	(5 sheets)
App. 1c	Part 3	'LPPSR21'	(7 sheets)

Appendix 2 Memory Matrix Listings

App. 2a	M.M Cell Addresses and Features for F5 Machine	(4 sheets)
App. 2b	M.M Cell Contents Trained to 'THIN' in Experiment 8	(2 sheets)
App. 2c	M.M Cell Contents Trained to 'CLEAN UP' in Experiment 11	(2 sheets)
App. 2d	M.M Cell Contents Trained to 'EDGE FIND' in Experiment 12	(2 sheets)

Assembly Listing of LPP Software Simulator

Part 1 - Control, Train/Test, Data Handler Sections

```
00010                    NAM    LPPF5
00020                    OPT    O,NOG,NOP
00030 0100               ORG    $0100
00040            *
00050            * THIS PROGRAM REQUIRES THE FOLLOWING
00060            * TO RUN CORRECTLY -
00070            * 'LPPSR21'
00080            * 'LIB1'
00090            * 'LPPMM1'
00100            *
00110     E07E   PDATA1 EQU    $E07E  External Equates
00120     131D   EDGE   EQU    $131D  for Routines,
00130     132B   ADDR   EQU    $132B      Variables,
00140     1318   XCO    EQU    $1318
00150     1319   YCO    EQU    $1319  in: LPPSR21
00160     11A2   GETWIN EQU    $11A2      ($1000-1900)
00170     1190   GETPO  EQU    $1190
00180     1199   PUTPO  EQU    $1199      LIB1
00190     E1AC   INEEE  EQU    $E1AC      ($A800-AA00)
00200     E0B1   WAIT2  EQU    $E0B1
00210     14B5   D2B2V  EQU    $14B5      LPPMM1
00220     131E   PO     EQU    $131E      ($3200-3600)
00230     131F   P1     EQU    $131F
00240     1320   P2     EQU    $1320      MSIBUG
00250     1321   P3     EQU    $1321      ($E000-E400)
00260     1322   P4     EQU    $1322      System Monitor
00270     1323   P5     EQU    $1323      in ROM
00280     1324   P6     EQU    $1324
00290     1325   P7     EQU    $1325
00300     1326   P8     EQU    $1326
00310     133A   SOURCE EQU    $133A
00320     133C   DEST   EQU    $133C
00330     133E   SRCNO  EQU    $133E
00340     133F   INCR   EQU    $133F
00350     1340   STS    EQU    $1340
00360     1342   DTS    EQU    $1342
00370     E047   BADDR  EQU    $E047
00380     E067   OUTHL  EQU    $E067
00390     E06B   OUTHR  EQU    $E06B
00400     1430   PPAT   EQU    $1430
00410     1332   PITO   EQU    $1332
00420     1333   PIT1   EQU    $1333
00430     1334   PITX   EQU    $1334
00440     1344   VDUNOI EQU    $1344
00450     1345   BUFNOI EQU    $1345
00460     1346   SRCI   EQU    $1346
00470     1348   STSI   EQU    $1348
00480     134A   SRCNOI EQU    $134A
00490     134B   INCRI  EQU    $134B
00500     0000   TRACK  EQU    $0000
00510     000C   NOSECT EQU    $000C
00520     134C   VDUNOE EQU    $134C
00530     134D   BUFNOE EQU    $134D
00540     134E   SRCE   EQU    $134E
00550     1350   STSE   EQU    $1350
00560     1352   SRCNOE EQU    $1352
00570     1353   INCRE  EQU    $1353
00580     2439   READ   EQU    $2439
00590     2758   WRITE  EQU    $2758
00600     000A   EMEMH  EQU    $000A
00610     1354   VDUNOO EQU    $1354
00620     1355   BUFNOO EQU    $1355
00630     1356   DESTO  EQU    $1356
00640     1358   DTSO   EQU    $1358
00650     135C   INCR1  EQU    $135C
00660     1700   PREP   EQU    $1700
00670     A800   INIT   EQU    $A800
00680     171C   INXTS  EQU    $171C
00690     1731   SINIT1 EQU    $1731
00700     1772   SINIT2 EQU    $1772
00710     177F   SINIT3 EQU    $177F
00720     1797   DINIT  EQU    $1797
00730     132E   BITO   EQU    $132E
00740     132F   BIT1   EQU    $132F
00750     135F   BITX   EQU    $135F
00760     1800   GIPPP  EQU    $1800
00770     1819   GEXPP  EQU    GIPPP+$19
00780     1830   GOPPP  EQU    GIPPP+$30
00790     1843   SIPPP  EQU    GIPPP+$43
00800     185C   SEXPP  EQU    GIPPP+$5C
00810     1875   SOPPP  EQU    GIPPP+$75
00820     E089   INHEX  EQU    $E089
00830     1713   GETTS  EQU    $1713
00840     1300   VDUNO  EQU    $1300
00850     1301   BUFNO  EQU    $1301
00860     A910   CLRVDU EQU    $A910
00870     F502   TTYIN  EQU    $F502
00880     112E   XYLOC  EQU    $112E
00890     4000   MMST   EQU    $4000
00900     E055   BYTE   EQU    $E055
00910     1337   PRTON  EQU    $1337
00920     1338   PRTOFF EQU    $1338
00930     E0CC   OUTS   EQU    $E0CC
00940     E1D1   OUTEEE EQU    $E1D1
00950     3200   MMHAND EQU    $3200
00960     3291   RANDR  EQU    $3291
00970            *
00980 0100 CE 0380 LOOP   LDX    £XHEAD    - START OF CONTROL
00990 0103 BD E07E        JSR    PDATA1    - LOOP
01000 0106 BD 065F        JSR    CLEAR      "LPP System..."
01010 0109 BD A800        JSR    INIT       Initialize I/Fs
01020 010C CE 03AD LOOP1  LDX    £XCL1
01030 010F BD 1700        JSR    PREP       "Control ?"
01040 0112 C1 44          CMP B  £'D
01050 0114 26 05          BNE    *+7        Is it D ?
01060 0116 BD 012F        JSR    DHAND      Yes,Datahandler..
01070 0119 20 F1          BRA    LOOP1
01080 011B C1 54          CMP B  £'T
```

Assembly Listing of LPP Software Simulator

Part 1 - Control, Train/Test, Data Handler Sections

```
01090 011D 26 05          BNE    *+7        Is it T ?
01100 011F BD 06AB        JSR    TTRUN      Yes, T/Trun...
01110 0122 20 E8          BRA    LOOP1
01120 0124 C1 4D          CMP B  £'M
01130 0126 26 05          BNE    *+7        Is it M ?
01140 0128 BD 3200        JSR    MMHAND     Yes, MMhandler...
01150 012B 20 DF          BRA    LOOP1
01160 012D 20 DD          BRA    LOOP1      Illegal - ignore
01170                *
01180 012F CE 03F0 DHAND  LDX    £XDH1    - DATA HANDLER LOOP
01190 0132 BD 1700        JSR    PREP       "Data Handler ?"
01200 0135 C1 43          CMP B  £'C
01210 0137 26 01          BNE    *+3        Is it C ?
01220 0139 39             RTS               Return to Control
01230 013A C1 4D          CMP B  £'M
01240 013C 26 05          BNE    *+7        Is it M ?
01250 013E BD 0160        JSR    MOVED      Yes,Move data...
01260 0141 20 EC          BRA    DHAND
01270 0143 C1 44          CMP B  £'D
01280 0145 26 05          BNE    *+7        Is it D ?
01290 0147 BD 0600        JSR    DISPD      Yes,Displaydata..
01300 014A 20 E3          BRA    DHAND
01310 014C C1 45          CMP B  £'E
01320 014E 26 05          BNE    *+7        Is it E ?
01330 0150 BD 09B1        JSR    EDITD      Yes,Edit data...
01340 0153 20 DA          BRA    DHAND
01350 0155 C1 50          CMP B  £'P
01360 0157 26 05          BNE    *+7        Is it P ?
01370 0159 BD 0960        JSR    PRINTD     Yes,Print data...
01380 015C 20 D1          BRA    DHAND
01390 015E 20 CF          BRA    DHAND      Illegal - ignore
01400                *
01410 0160 CE 03E5 MOVED  LDX    £XMD1    - MOVE DATA BETWEEN
01420 0163 BD 1731        JSR    SINIT1   - BULK STORAGE MEDIA
01430 0166 BD 1772        JSR    SINIT2
01440 0169 BD 177F        JSR    SINIT3     Initialize Source
01450 016C 7F 1300        CLR    VDUNO      (as IPP)
01460 016F BD 1843        JSR    SIPPP      Specify L.H. VDU
01470 0172 CE 021A        LDX    £DUMMY
01480 0175 FF 134E        STX    SRCE       Dummy Source of
01490 0178 7F 1353        CLR    INCRE       for EXP
01500 017B 7C 1353        INC    INCRE
01510 017E CE 03E5        LDX    £XMD1
01520 0181 BD 1797        JSR    DINIT      Initialize Desti-
01530 0184 BD 1875        JSR    SOPPP       nation (as OPP)
01540 0187 CE 021A        LDX    £DUMMY
01550 018A FF 0301        STX    IOFUNC     Dummy function
01560 018D 7F 0300        CLR    PAUSE      Specify NO pauses
01570 0190 7F 030D        CLR    PASSES
01580 0193 7C 030D        INC    PASSES     Specify ONE pass..
01590 0196 7E 0199        JMP    IOLOOP     Perform Transfer
01600                *
01610 0199 CE 04E0 IOLOOP LDX    £XREADY  - TO PERFORM IP/OP
01620 019C BD 1700        JSR    PREP     - TRANSFERS AND
01630 019F B6 030D        LDA A  PASSES   - 'JSR IOFUNC'
01640 01A2 B7 030E        STA A  PASSNO
01650 01A5 BD 1800        JSR    GIPPP      Get IPP Parameters
01660 01A8 BD 0225        JSR    ITRAN      Do I/P transfer
01670 01AB BD 1819        JSR    GEXPP      Get EXP Parameters
01680 01AE BD 0225        JSR    ITRAN      Do I/P transfer
01690 01B1 FE 0301 IO4    LDX    IOFUNC     Get Function
01700 01B4 AD 00          JSR    0,X        Perform Function
01710 01B6 7A 030E        DEC    PASSNO
01720 01B9 26 60          BNE    IO5        Passes finished ?
01730 01BB BD 1830        JSR    GOPPP      Get OPP Parameters
01740 01BE BD 022F        JSR    OTRAN      Do O/P transfer
01750 01C1 BD 08BF        JSR    DPAUSE     Pause, if told to
01760 01C4 7A 134A        DEC    SRCNOI     End of Source
01770 01C7 27 51          BEQ    DUMMY       file ?
01780 01C9 FE 1358        LDX    DTSO
01790 01CC BD 171C        JSR    INXTS      Increment Disk T+S
01800 01CF FF 1358        STX    DTSO        for OPP File by 1
01810 01D2 B6 134B        LDA A  INCRI      Store IPP File
01820 01D5 B7 135C        STA A  INCR1       increment
01830 01D8 FE 1348 IO3    LDX    STSI
01840 01DB BD 171C        JSR    INXTS      Increment Disk T+S
01850 01DE FF 1348        STX    STSI        for IPP File by
01860 01E1 7A 135C        DEC    INCR1       'INCRI'
01870 01E4 27 10          BEQ    IO2
01880 01E6 FE 1346        LDX    SRCI       Is IPP Source a
01890 01E9 8C 14B5        CPX    £D2B2V      disk file ?
01900 01EC 27 EA          BEQ    IO3
01910 01EE BD 1800        JSR    GIPPP      No,so do an I/P
01920 01F1 BD 0225        JSR    ITRAN       transfer to skip
01930 01F4 20 E2          BRA    IO3         unwanted patterns
01940 01F6 B6 1353 IO2    LDA A  INCRE      Store EXP File
01950 01F9 B7 135C        STA A  INCR1       increment
01960 01FC FE 1350        LDX    STSE
01970 01FF BD 171C        JSR    INXTS      Increment Disk T+S
01980 0202 FF 1350        STX    STSE        for EXP File by
01990 0205 7A 135C        DEC    INCR1       'INCRE'
02000 0208 27 95          BEQ    IOLOOP+6
02010 020A FE 134E        LDX    SRCE       Is EXP Source a
02020 020D 8C 14B5        CPX    £D2B2V      disk file ?
02030 0210 27 EA          BEQ    IO2+6
02040 0212 BD 1819        JSR    GEXPP      No, so do an I/P
02050 0215 BD 0225        JSR    ITRAN       transfer to skip
02060 0218 20 E2          BRA    IO2+6       unwanted patterns
02070 021A 39       DUMMY RTS               (Dummy Function)
02080 021B 7D 1354 IO5    TST    VDUNOO     If >1 pass, copy
02090 021E 27 91          BEQ    IO4         pattern if in
02100 0220 BD 09F6        JSR    COPY10      'parallel mode'
02110 0223 20 8C          BRA    IO4         from R>L.H. VDU
02120                *
02130 0225 FE 1340 ITRAN  LDX    STS      - DO I/P TRANSFER
02140 0228 DF 00          STX    TRACK    - 'JSR SOURCE'
02150 022A FE 133A        LDX    SOURCE
02160 022D 6E 00          JMP    0,X
```

Assembly Listing of LPP Software Simulator

Part 1 - Control, Train/Test, Data Handler Sections

```
02170                   *
02180 022F FE 1342 OTRAN  LDX    DTS       - DO O/P TRANSFER
02190 0232 DF 00          STX    TRACK     - 'JSR DEST'
02200 0234 FE 133C        LDX    DEST
02210 0237 6E 00          JMP    0,X
02220                   *
02230 0300                ORG    $0300
02240 0300 0001   PAUSE   RMB    1          Variable Table:
02250 0301 0002   IOFUNC  RMB    2
02260 0303 0002   XYFUNC  RMB    2           - flags,
02270 0305 02     NOMMS   FCB    $02         - temporary
02280 0306 0C     FFEED   FCB    $0C           stores,
02290 0307 0001   PERM0   RMB    1           - constants.
02300 0308 0001   PERM1   RMB    1
02310 0309 0001   TEMP0   RMB    1
02320 030A 0001   TEMP1   RMB    1
02330 030B 0001   COUNT1  RMB    1
02340 030C 0001   COUNT2  RMB    1
02350 030D 0001   PASSES  RMB    1
02360 030E 0001   PASSNO  RMB    1
                                            Text Strings:
02370                   *
02380 0380                ORG    $0380
02390 0380 0D     XHEAD   FCB    $0D,$0A,$0A
02400 0383 4C             FCC    'LPP SYSTEM 10 -
                                  MEMORY MATRIX FORMAT '
02410 03A8 46             FCC    'F5'
02420 03AA 0D             FCB    $0D,$0A,$04
02430 03AD 43     XCL1    FCC    'CONTROL LOOP -
                                  DATA HANDLER , T/T RUN , '
02440 03D5 4D             FCC    'M.M. HANDLER ? '
02450 03E4 04             FCB    4
02460 03E5 44     XMD1    FCC    'DATA MOVER'
02470 03EF 04             FCB    4
02480 03F0 44     XDH1    FCC    'DATA HANDLER - '
02490 03FF 43             FCC    'CONTROL , MOVE , DISPLAY ,
                                  EDIT , PRINT DATA ? '
02500 042E 04             FCB    4
02510 042F 44     XDD2    FCC    'DATA II DISPLAYER'
02520 0440 04             FCB    4
02530 0441 44     XDD1    FCC    'DATA I DISPLAYER'
02540 0451 04             FCB    4
02550 0452 44     XDP1    FCC    'DATA I PRINTER'
02560 0460 04             FCB    4
02570 0461 44     XDP2    FCC    'DATA II PRINTER'
02580 0470 04             FCB    4
02590 0471 44     XED1    FCC    'DATA EDITOR'
02600 047C 04             FCB    4
02610 047D 56     XCLEAR  FCC    'VDU TO BE CLEARED
                                  ("Y" OR "N") ? '
02620 049E 04             FCB    4
02630 049F 45     XGEDGE  FCC    'EDGE POINTS ("1" OR "0")?'
02640 04BA 04             FCB    4
02650 04BB 50     XPMM1   FCC    'PRESET MEMORY MATRIX
                                  ("Y" OR "N") ? '
02660 04DF 04             FCB    4
02670 04E0 52     XREADY  FCC    'READY (X) ? '
02680 04EC 04             FCB    4
02690 04ED 4E     XNPASS  FCC    'NO. OF PASSES (HH) ? '
02700 0502 04             FCB    4
02710                   *
02720 0600                ORG    $0600
02730 0600 CE 0441 DISPD  LDX    £XDD1     - DISPLAY
02740 0603 BD 1731        JSR    SINIT1    - CHARACTERS
02750 0606 BD 1772        JSR    SINIT2    "Displayer 1 ?"
02760 0609 BD 177F        JSR    SINIT3    Initialize Source
02770 060C 7F 1300        CLR    VDUNO     Specify L.H. VDU
02780 060F BD 1843        JSR    SIPPP     Store Parameters
02790 0612 CE 042F        LDX    £XDD2
02800 0615 BD 1731        JSR    SINIT1    "Displayer 2 ?"
02810 0618 BD 177F        JSR    SINIT3    Initialize Source
02820 061B 7C 1300        INC    VDUNO     Specify R.H. VDU
02830 061E BD 185C        JSR    SEXPP     Store Parameters
02840 0621 CE 021A        LDX    £DUMMY
02850 0624 FF 1356        STX    DESTO     Dummy Destination
02860 0627 CE 0639        LDX    £DREADY   Put 'Ready ?'
02870 062A FF 0301        STX    IOFUNC     function in loop
02880 062D 7F 0300        CLR    PAUSE     Set m/c to Pause
02890 0630 7F 030D        CLR    PASSES     between patterns
02900 0633 7C 030D        INC    PASSES    Specify ONE pass..
02910 0636 7E 0199        JMP    IOLOOP    Start displaying..
02920                   *
02930 0639 CE 04E0 DREADY LDX    £XREADY   - DO A READY PAUSE
02940 063C 7E 1700        JMP    PREP      "Ready ? "
02950                   *
02960 063F 7F 1318 XYLOOP CLR    XCO       - 'JSR XYFUNC' FOR
02970 0642 7F 1319        CLR    YCO       - EACH X AND Y IN
02980 0645 FE 0303        LDX    XYFUNC    - RANGE $00-$1F
02990 0648 AD 00          JSR    0,X       Do function
03000 064A 7C 1319        INC    YCO
03010 064D B6 1319        LDA A  YCO
03020 0650 81 20          CMP A  £$20      Y finished ?
03030 0652 26 F1          BNE    XYLOOP+6
03040 0654 7C 1318        INC    XCO
03050 0657 B6 1318        LDA A  XCO
03060 065A 81 20          CMP A  £$20      X finished ?
03070 065C 26 E4          BNE    XYLOOP+3
03080 065E 39             RTS
03090                   *
03100 065F CE 047D CLEAR  LDX    £XCLEAR   - CLEAR VDU IF
03110 0662 BD 1700        JSR    PREP      - REQUIRED
03120 0665 C1 4E          CMP B  £'N       "Clear ? "
03130 0667 27 07          BEQ    CLEAR1    Is it N ?
03140 0669 C1 59          CMP B  £'Y       Is it Y ?
03150 066B 26 F2          BNE    CLEAR     Illegal - ignore
03160 066D 7E A910        JMP    CLRVDU    Clear VDU
03170 0670 39     CLEAR1  RTS
03180                   *
```

Assembly Listing of LPP Software Simulator

Part 1 - Control, Train/Test, Data Handler Sections

```
03190 0671 CE 048B PREMM  LDX    £XPMM1    - PRESET MEMORY
03200 0674 BD 1700        JSR    PREP      - MATRIX IF
03210 0677 C1 4E          CMP B  £'N       - REQUIRED
03220 0679 27 15          BEQ    PMM1      "Preset ? "
03230 067B C1 59          CMP B  £'Y       Is it N / Y ?
03240 067D 26 F2          BNE    PREMM     Illegal - ignore
03250 067F B6 135F        LDA A  BITX      Yes, put 'BITX'
03260 0682 CE 4000        LDX    £MMST      in every M.M.
03270 0685 A7 00          STA A  0,X        location....
03280 0687 08             INX
03290 0688 8C 4200        CPX    £MMST+$200
03300 068B 26 F8          BNE    *-6          Finished ?
03310 068D 7E E081        JMP    WAIT2        Do CRLF...
03320 0690 39       PMM1  RTS
03330               *
03340 0691 CE 049F GEDGE  LDX    £XGEDGE   - GET EDGE POINT
03350 0694 BD 1700        JSR    PREP      - VALUE
03360 0697 C1 31          CMP B  £'1       "Edge ?" Is it 1 ?
03370 0699 27 09          BEQ    *+11
03380 069B C1 30          CMP B  £'0       Is it 0 ?
03390 069D 26 F2          BNE    GEDGE     Illegal - ignore
03400 069F F6 132E        LDA B  BIT0      Put 'BIT0',
03410 06A2 20 03          BRA    *+5
03420 06A4 F6 132F        LDA B  BIT1       or 'BIT1'
03430 06A7 F7 131D        STA B  EDGE       into 'EDGE'
03440 06AA 39             RTS
03450               *
03460 06AB CE 0700 TTRUN  LDX    £XTTR1    - T/T RUN
03470 06AE BD 1700        JSR    PREP      "T/Trun ? "
03480 06B1 C1 54          CMP B  £'T       Is it T ?
03490 06B3 27 06          BEQ    JTEST
03500 06B5 C1 4E          CMP B  £'N       Is it N ?
03510 06B7 26 F2          BNE    *-12      Illegal - ignore
03520 06B9 20 03          BRA    TRAIN
03530 06BB 7E 0833 JTEST  JMP    TEST
03540 06BE CE 0736 TRAIN  LDX    £XTTR3    - TRAINING RUN
03550 06C1 BD 1731        JSR    SINIT1    "IPP Source ? "
03560 06C4 BD 1772        JSR    SINIT2    Initialize IPP
03570 06C7 BD 177F        JSR    SINIT3     file
03580 06CA 7F 1300        CLR    VDUNO     Specify L.H. VDU
03590 06CD BD 1843        JSR    SIPPP     Store parameters
03600 06D0 CE 074A        LDX    £XTTR4    "EXP Source ? "
03610 06D3 BD 1731        JSR    SINIT1    Initilaize EXP
03620 06D6 BD 177F        JSR    SINIT3     file
03630 06D9 7C 1300        INC    VDUNO     Specify R.H. VDU
03640 06DC BD 185C        JSR    SEXPP     Store parameters
03650 06DF CE 021A        LDX    £DUMMY    Set 'Dummy'
03660 06E2 FF 1356        STX    DESTO      for destination
03670 06E5 CE 0872        LDX    £LEARN    Set 'Learn'
03680 06E8 FF 0301        STX    IOFUNC     in loop
03690 06EB 7F 0300        CLR    PAUSE     NO pauses...
03700 06EE BD 0691        JSR    GEDGE     Get edge value
03710 06F1 BD 0671        JSR    PREMM     Preset M.M.
03720 06F4 7F 030D        CLR    PASSES    Specify ONE
03730 06F7 7C 030D        INC    PASSES     pass in each loop
03740 06FA BD 0199        JSR    IOLOOP    Enter loop
03750 06FD 7E 0809        JMP    SMM       Store M.M.....
```

More Text Strings:

```
03760               *
03770 0700 54       XTTR1 FCC    'T/T RUN SECTION - '
03780 0712 0D             FCB    $0D,$0A
03790 0714 54             FCC    'TRAIN OR TEST RUN
                                  ("N" OR "T") ? '
03800 0735 04             FCB    4
03810 0736 49       XTTR3 FCC    'INPUT PATTERN (IPP)'
03820 0749 04             FCB    4
03830 074A 45       XTTR4 FCC    'EXAMPLE PATTERN (EXP) '
03840 0760 04             FCB    4
03850 0761 4F       XTTR5 FCC    'OUTPUT PATTERN (OPP)'
03860 0775 04             FCB    4
03870 0776 4C       XLMM1 FCC    'LOAD MEMORY MATRIX
                                  ("Y" OR "N") ? '
03880 0798 04             FCB    4
03890 0799 53       XSMM1 FCC    'STORE MEMORY MATRIX
                                  ("Y" OR "N") ? '
03900 07BC 04             FCB    4
03910 07BD 50       XSETP1 FCC   'PAUSE AFTER EACH PATTERN
                                  ("Y" OR "N") ? '
03920 07E5 04             FCB    4
03930 07E6 53       XSORP FCC    'SERIAL OR PARALLEL
                                  ("S" OR "P") ? '
03940 0808 04             FCB    4
03950               *
03960 0809 CE 0799 SMM    LDX    £XSMM1    - STORE M.M ON
03970 080C BD 1700        JSR    PREP      - DISK IF REQUIRED
03980 080F C1 4E          CMP B  £'N       "Store ?"
03990 0811 27 1F          BEQ    SMM1      Is it N ?
04000 0813 C1 59          CMP B  £'Y       Is it Y ?
04010 0815 26 F2          BNE    SMM       Illegal - ignore
04020 0817 B6 0305        LDA A  NOMMS     Get no. sectors
04030 081A 97 0C          STA A  NOSECT     needed on disk
04040 081C BD 1713        JSR    GETTS     Get T+S from
04050 081F DF 00          STX    TRACK      operator
04060 0821 CE 04E0        LDX    £XREADY   "Ready ? "
04070 0824 BD 1700        JSR    PREP
04080 0827 CE FFFF        LDX    £$FFFF    Initialize DOS
04090 082A DF 0A          STX    EMEMH      Routines
04100 082C CE 4000        LDX    £MMST
04110 082F 7E 2758        JMP    WRITE     Write M.M. to disk
04120 0832 39       SMM1  RTS
04130               *
04140 0833 CE 0736 TEST   LDX    £XTTR3    - TESTING RUN
04150 0836 BD 1731        JSR    SINIT1    "IPP Source ? "
04160 0839 BD 1772        JSR    SINIT2    Initilaize IPP
04170 083C BD 177F        JSR    SINIT3     file
04180 083F 7F 1300        CLR    VDUNO     Specify L.H. VDU
04190 0842 BD 1843        JSR    SIPPP     Store parameters
```

Assembly Listing of LPP Software Simulator

Part 1 - Control, Train/Test, Data Handler Sections

```
04200 0845 CE 021A          LDX    £DUMMY    Set 'Dummy' for
04210 0848 FF 134E          STX    SRCE       EXP source
04220 084B 7F 1353          CLR    INCRE
04230 084E 7C 1353          INC    INCRE
04240 0851 CE 0761          LDX    £XTTR5    "OPP Destination?"
04250 0854 BD 1797          JSR    DINIT     Init.OPP file
04260 0857 BD 099C          JSR    SORP      Serial or parallel
04270 085A BD 1875          JSR    SOPPP     Store parameters
04280 085D BD 08AA          JSR    SETPS     Pauses ?
04290 0860 BD 0691          JSR    GEDGE     Get edge value
04300 0863 BD 0880          JSR    LMM       Load M.M. ?
04310 0866 CE 087B          LDX    £WORK     Set 'Work'
04320 0869 FF 0301          STX    IOFUNC     in loop
04330 086C BD 083D          JSR    NPASS     No. passes ?
04340 086F 7E 0199          JMP    IOLOOP    Enter loop
04350                 *
04360 0872 CE 08CB LEARN    LDX    £LEARN1   - SCAN PATTERN AND
04370 0875 FF 0303          STX    XYFUNC    - 'JSR LEARN1' AT
04380 0878 7E 063F          JMP    XYLOOP    - EACH POINT
04390                 *
04400 087B CE 08EC WORK     LDX    £WORK1    - SCAN PATTERN AND
04410 087E 20 F5            BRA    LEARN+3   - 'JSR WORK1' AT
04420                 *                      - EACH POINT

04430 0880 CE 0776 LMM      LDX    £XLMM1    - LOAD M.M. FROM
04440 0883 BD 1700          JSR    PREP      - DISK > CORE
04450 0886 C1 4E            CMP B  £'N       - IF REQUIRED
04460 0888 27 1F            BEQ    LMM1      "Load M.M. ? "
04470 088A C1 59            CMP B  £'Y       Is it Y/N ?
04480 088C 26 F2            BNE    LMM       Illegal - ignore
04490 088E B6 0305          LDA A  NOMMS     Get no. of sectors
04500 0891 97 0C            STA A  NOSECT     needed on disk
04510 0893 BD 1713          JSR    GETTS     Get T+S from
04520 0896 DF 00            STX    TRACK      operator
04530 0898 CE 04E0          LDX    £XREADY   "Ready ? "
04540 089B BD 1700          JSR    PREP
04550 089E CE FFFF          LDX    £$FFFF    Initilaize DOS
04560 08A1 DF 0A            STX    EMEMH      routines
04570 08A3 CE 4000          LDX    £MMST
04580 08A6 7E 2439          JMP    READ      Read M.M.from disk
04590 08A9 39       LMM1    RTS
04600                 *
04610 08AA 7F 0300 SETPS    CLR    PAUSE     - REQUEST 'PAUSE
04620 08AD CE 07BD          LDX    £XSETP1   - OR NOT ? '
04630 08B0 BD 1700          JSR    PREP      "Pause between
04640 08B3 C1 4E            CMP B  £'N        patterns ? "
04650 08B5 27 07            BEQ    *+9       Is it Y/N ?
04660 08B7 C1 59            CMP B  £'Y
04670 08B9 26 EF            BNE    SETPS     Illegal - ignore
04680 08BB 73 0300          COM    PAUSE     Set pause flag
04690 08BE 39               RTS
04700                 *
04710 08BF 7D 0300 DPAUSE   TST    PAUSE     - DO A PAUSE IF
04720 08C2 27 06            BEQ    *+8       - PAUSE FLAG SET
04730 08C4 CE 04E0          LDX    £XREADY   "Ready ? "
04740 08C7 7E 1700          JMP    PREP      Accept any char.
04750 08CA 39               RTS
04760                 *
04770 08CB B6 1344 LEARN1   LDA A  VDUNOI    - LEARN IN MM F5
04780 08CE B7 1300          STA A  VDUNO     - FROM IPP+EXP
04790 08D1 BD 11A2          JSR    GETWIN    Get window at x,y
04800 08D4 BD 0909          JSR    CADDR      from IPP
04810 08D7 B6 134C          LDA A  VDUNOE    Calculate MM addr.
04820 08DA B7 1300          STA A  VDUNO
04830 08DD BD 1190          JSR    GETPO     Get EXP/PO at x,y
04840 08E0 FE 132B          LDX    ADDR
04850 08E3 B6 131E          LDA A  PO
04860 08E6 A7 00            STA A  0,X       Store in MM cell
04870 08E8 BD 3291          JSR    RANDR     Rot+Ref and store
04880 08EB 39               RTS               in other MM cells
04890                 *
04900 08EC B6 1344 WORK1    LDA A  VDUNOI - WORK USING MM F5
04910 08EF B7 1300          STA A  VDUNO  - ON IPP TO GIVE OPP
04920 08F2 BD 11A2          JSR    GETWIN    Get window at x,y
04930 08F5 BD 0909          JSR    CADDR      from IPP
04940 08F8 FE 132B          LDX    ADDR      Calculate MM addr.
04950 08FB A6 00            LDA A  0,X       Get MM data from
04960 08FD B7 131E          STA A  PO         that cell
04970 0900 B6 1354          LDA A  VDUNOO
04980 0903 B7 1300          STA A  VDUNO     Put into OPP/PO
04990 0906 7E 1199          JMP    PUTPO      at x,y
05000                 *
05010 0909 CE 4000 CADDR    LDX    £MMST  - CALCULATE ADDR FOR
05020 090C FF 132B          STX    ADDR   - MM F5 FROM WINDOW
05030 090F B6 131E          LDA A  PO       Get PO
05040 0912 84 01            AND A  £1       Mask to Bit0
05050 0914 F6 131F          LDA B  P1       Get P1
05060 0917 C4 01            AND B  £1       Mask/shift to Bit1
05070 0919 58               ASL B
05080 091A 1B               ABA
05090 091B F6 1320          LDA B  P2       Get P2
05100 091E C4 01            AND B  £1       Mask/shift to Bit2
05110 0920 58               ASL B
05120 0921 58               ASL B
05130 0922 1B               ABA
05140 0923 F6 1321          LDA B  P3       Get P3
05150 0926 C4 01            AND B  £1       Mask/shift to Bit3
05160 0928 58               ASL B
05170 0929 58               ASL B
05180 092A 58               ASL B
05190 092B 1B               ABA
05200 092C F6 1322          LDA B  P4       Get P4
05210 092F C4 01            AND B  £1       Mask/shift to Bit4
05220 0931 58               ASL B
05230 0932 58               ASL B
05240 0933 58               ASL B
05250 0934 58               ASL B
05260 0935 1B               ABA
```

Assembly Listing of LPP Software Simulator

Part 1 - Control, Train/Test, Data Handler Sections

```
05270 0936 F6 1323          LDA B  P5       Get P5
05280 0939 C4 01            AND B  £1       Mask/shift to Bit5
05290 093B 56               ROR B
05300 093C 56               ROR B
05310 093D 56               ROR B
05320 093E 56               ROR B
05330 093F 1B               ABA
05340 0940 F6 1324          LDA B  P6       Get P6
05350 0943 C4 01            AND B  £1       Mask/shift to Bit6
05360 0945 56               ROR B
05370 0946 56               ROR B
05380 0947 56               ROR B
05390 0948 1B               ABA
05400 0949 F6 1325          LDA B  P7       Get P7
05410 094C C4 01            AND B  £1       Mask/shift to Bit7
05420 094E 56               ROR B
05430 094F 56               ROR B
05440 0950 1B               ABA             Store as LS Digit
05450 0951 B7 132C          STA A  ADDR+1    of MM address
05460 0954 B6 1326          LDA A  P8       Get P8
05470 0957 84 01            AND A  £1       Mask to Bit0
05480 0959 BA 132B          ORA A  ADDR     OR with MS Digit
05490 095C B7 132B          STA A  ADDR      of MM address,
05500 095F 39               RTS              now in 'ADDR'
05510                 *
05520 0960 CE 0452 PRINTD   LDX    £XDP1    - PRINT CHARACTERS
05530 0963 BD 1731          JSR    SINIT1   "Printer 1 ? "
05540 0966 BD 1772          JSR    SINIT2   Initilaize source
05550 0969 BD 177F          JSR    SINIT3
05560 096C 7F 1300          CLR    VDUNO    Specify L.H. VDU
05570 096F BD 1843          JSR    SIPPP    Store parameters
05580 0972 CE 0461          LDX    £XDP2    "Printer 2 ? "
05590 0975 BD 1731          JSR    SINIT1   Initialize source
05600 0978 BD 177F          JSR    SINIT3
05610 097B 7C 1300          INC    VDUNO    Specify R.H. VDU
05620 097E BD 185C          JSR    SEXPP    Store paramters
05630 0981 CE 021A          LDX    £DUMMY   Set 'Dummy'
05640 0984 FF 1356          STX    DESTO     for destination
05650 0987 CE 1430          LDX    £PPAT    Set 'Print'
05660 098A FF 0301          STX    IOFUNC    in loop
05670 098D 7F 0300          CLR    PAUSE    No pauses....
05680 0990 BD 0639          JSR    DREADY   "Ready ? "
05690 0993 7F 030D          CLR    PASSES
05700 0996 7C 030D          INC    PASSES   Set one pass...
05710 0999 7E 0199          JMP    IOLOOP   Enter loop...
05720                 *
05730 099C 7F 1300 SORP     CLR    VDUNO    - SET VDUNO FROM
05740 099F CE 07E6          LDX    £XSORP   - SERIAL OR
05750 09A2 BD 1700          JSR    PREP     - PARALLEL MODE
05760 09A5 C1 53            CMP B  £'S      "Ser.or Par.? "
05770 09A7 27 07            BEQ    SORP1    Is it S/P ? "
05780 09A9 C1 50            CMP B  £'P
05790 09AB 26 EF            BNE    SORP     Illegal - ignore
05800 09AD 7C 1300          INC    VDUNO    Set R.H. VDU if
05810 09B0 39       SORP1   RTS              parallel mode
05820               *
05830 09B1 CE 0471  EDITD   LDX    £XED1    - EDIT PICTURES
05840 09B4 BD 1731          JSR    SINIT1   - ON VDU
05850 09B7 BD 1772          JSR    SINIT2   "Editor ? "
05860 09BA BD 177F          JSR    SINIT3   Initilaize source
05870 09BD 7F 1300          CLR    VDUNO     as IPP on L.H.VDU
05880 09C0 BD 1843          JSR    SIPPP    Store parameters
05890 09C3 CE 021A          LDX    £DUMMY
05900 09C6 FF 134E          STX    SRCE     Set 'dummy'
05910 09C9 7F 1353          CLR    INCRE     for EXP
05920 09CC 7C 1353          INC    INCRE
05930 09CF CE 0471          LDX    £XED1
05940 09D2 BD 1797          JSR    DINIT    Initialize
05950 09D5 7C 1300          INC    VDUNO     destination
05960 09D8 BD 1875          JSR    SOPPP    Store parameters
05970 09DB CE 0A1F          LDX    £EDFUNC  Set 'Edit'
05980 09DE FF 0301          STX    IOFUNC    in loop
05990 09E1 7F 0300          CLR    PAUSE    NO pauses...
06000 09E4 7F 030D          CLR    PASSES
06010 09E7 7C 030D          INC    PASSES   ONE pass.....
06020 09EA 7E 0199          JMP    IOLOOP   Enter loop....
06030               *
06040 09ED CE 09FB COPY01   LDX    £C01     - COPY PICTURE
06050 09F0 FF 0303          STX    XYFUNC   - FROM VDU0 > VDU1
06060 09F3 7E 063F          JMP    XYLOOP          (L.H.) R.H.)
06070               *
06080 09F6 CE 0A0E COPY10   LDX    £C10     - COPY PICTURE
06090 09F9 20 F5            BRA    COPY01+3 - FROM VDU1 > VDU0
06100               *                               (R.H.) L.H.)
06110               *
06120 09FB 7F 1300 C01      CLR    VDUNO    - COPY PIXEL AT X,Y
06130 09FE BD 112E          JSR    XYLOC    - FROM VDU0 > VDU1
06140 0A01 A6 00            LDA A  0,X       Get pixel addr.
06150 0A03 36               PSH A            on L.H.VDU and
06160 0A04 7C 1300          INC    VDUNO     get pixel
06170 0A07 BD 112E          JSR    XYLOC    Get pixel addr.
06180 0A0A 32               PUL A            on R.H.VDU and
06190 0A0B A7 00            STA A  0,X       store pixel
06200 0A0D 39               RTS
06210               *
06220 0A0E 7F 1300 C10      CLR    VDUNO    - COPY PIXEL AT X,Y
06230 0A11 7C 1300          INC    VDUNO    - FROM VDU1 > VDU0
06240 0A14 BD 112E          JSR    XYLOC    Get pixel addr.
06250 0A17 A6 00            LDA A  0,X       on R.H.VDU and
06260 0A19 36               PSH A            get pixel
06270 0A1A 7F 1300          CLR    VDUNO    Specify L.H.VDU
06280 0A1D 20 E8            BRA    C01+12    and go to C01+12
06290               *
06300 0A1F BD 09ED EDFUNC   JSR    COPY01   - EDIT PICTURES ON
06310 0A22 86 10            LDA A  £$10     - VDU0 > VDU1
06320 0A24 B7 1318          STA A  XC0      Copy pattern L>R
06330 0A27 B7 1319          STA A  YC0      Set x,y = centre
06340 0A2A 7F 1300 EDF1     CLR    VDUNO
```

Assembly Listing of LPP Software Simulator

Part 1 - Control, Train/Test, Data Handler Sections

```
06350 0A2D BD 112E        JSR    XYLOC     Get pixel at x,y
06360 0A30 A6 00          LDA A  0,X        from LH VDU
06370 0A32 B7 0307        STA A  PERM0     Store in 'PERM0'
06380 0A35 7C 1300        INC    VDUNO
06390 0A38 BD 112E        JSR    XYLOC     Get pixel at x,y
06400 0A3B A6 00          LDA A  0,X        from RH VDU
06410 0A3D B7 0308        STA A  PERM1     Store in 'PERM1'
06420 0A40 86 20          LDA A  £$20
06430 0A42 B7 0309        STA A  TEMP0     Put 'space' in
06440 0A45 B7 030A        STA A  TEMP1      'TEMP0+1'
06450 0A48 7F 030B        CLR    COUNT1    Initilaize
06460 0A4B 7F 030C        CLR    COUNT2     software counters
06470 0A4E 7F 1300 EDF2   CLR    VDUNO
06480 0A51 BD 112E        JSR    XYLOC     On LH VDU :0'
06490 0A54 A6 00          LDA A  0,X        swap 'TEMP0'
06500 0A56 F6 0309        LDA B  TEMP0      and pixel at x,y
06510 0A59 E7 00          STA B  0,X
06520 0A5B B7 0309        STA A  TEMP0
06530 0A5E 7C 1300        INC    VDUNO
06540 0A61 BD 112E        JSR    XYLOC     On RH VDU :
06550 0A64 A6 00          LDA A  0,X        swap 'TEMP1'
06560 0A66 F6 030A        LDA B  TEMP1      and pixel at x,y
06570 0A69 E7 00          STA B  0,X
06580 0A6B B7 030A        STA A  TEMP1
06590 0A6E B6 F502 EDF3   LDA A  TTYIN     Get control
06600 0A71 47             ASR A             terminal flag
06610 0A72 25 11          BCS    EDF4      Key pressed ?
06620 0A74 7C 030B        INC    COUNT1    No, increment
06630 0A77 26 F5          BNE    EDF3       count 1,try again
06640 0A79 7C 030C        INC    COUNT2    Increment count2,
06650 0A7C B6 030C        LDA A  COUNT2    Test count2=$40 ?
06660 0A7F 84 3F          AND A  £$3F
06670 0A81 26 EB          BNE    EDF3      No, try flag again
06680 0A83 20 C9          BRA    EDF2      Yes, flash pixels
06690 0A85 7F 1300 EDF4   CLR    VDUNO     Flag set, so :
06700 0A88 BD 112E        JSR    XYLOC     On LH VDU,
06710 0A8B B6 0307        LDA A  PERM0      restore 'PERM0'
06720 0A8E A7 00          STA A  0,X        to x,y
06730 0A90 7C 1300        INC    VDUNO
06740 0A93 BD 112E        JSR    XYLOC     On RH VDU,
06750 0A96 B6 0308        LDA A  PERM1      restore 'PERM1'
06760 0A99 A7 00          STA A  0,X        to x,y
06770 0A9B B6 F503        LDA A  TTYIN+1   Get character
06780 0A9E 84 7F          AND A  £$7F
06790 0AA0 81 45          CMP A  £'E       Is it E ?
06800 0AA2 26 01          BNE    *+3
06810 0AA4 39             RTS              Yes, End edit
06820 0AA5 81 30          CMP A  £'0       Is it < 0 ?
06830 0AA7 2D 65          BLT    EDF10
06840 0AA9 81 39          CMP A  £'9       No, is it > 9 ?
06850 0AAB 2E 61          BGT    EDF10
06860 0AAD 81 35          CMP A  £'5       No, is it 5 ?
06870 0AAF 26 1C          BNE    EDF6
06880 0AB1 7F 1300        CLR    VDUNO     Yes, on RH VDU :
06890 0AB4 7C 1300        INC    VDUNO
06900 0AB7 BD 112E        JSR    XYLOC     Get pixel at x,y
06910 0ABA A6 00          LDA A  0,X
06920 0ABC B1 132E        CMP A  BIT0      Is pixel '0' ?
06930 0ABF 27 05          BEQ    *+7
06940 0AC1 B6 132E        LDA A  BIT0      No,load '0'in acc.
06950 0AC4 20 03          BRA    *+5
06960 0AC6 B6 132F        LDA A  BIT1      Yes,load'1'in acc.
06970 0AC9 A7 00          STA A  0,X       Store acc. at x,y
06980 0ACB 20 0E          BRA    EDF7      (i.e invert pixel)
06990 0ACD 81 33   EDF6   CMP A  £'3       Is it > 3 ?
07000 0ACF 2E 16          BGT    EDF8
07010 0AD1 BD 0B27        JSR    INCXCO    No, increment x
07020 0AD4 81 31          CMP A  £'1       Is it 1 ?
07030 0AD6 26 06          BNE    *+8
07040 0AD8 BD 0B1E        JSR    DECYCO    Yes, decrement y
07050 0ADB 7E 0A2A EDF7   JMP    EDF1      Back to edit loop
07060 0ADE 81 33          CMP A  £'3       Is it 3 ?
07070 0AE0 26 F9          BNE    EDF7
07080 0AE2 BD 0B32        JSR    INCYCO    Yes, increment y
07090 0AE5 20 F4          BRA    EDF7
07100 0AE7 81 37   EDF8   CMP A  £'7       Is it < 7 ?
07110 0AE9 2D 15          BLT    EDF9
07120 0AEB BD 0B15        JSR    DECXCO    No, decrement x
07130 0AEE 81 37          CMP A  £'7       Is it 7 ?
07140 0AF0 26 05          BNE    *+7
07150 0AF2 BD 0B1E        JSR    DECYCO    Yes, decrement y
07160 0AF5 20 E4          BRA    EDF7
07170 0AF7 81 39          CMP A  £'9       Is it 9 ?
07180 0AF9 26 E0          BNE    EDF7
07190 0AFB BD 0B32        JSR    INCYCO    Yes, increment y
07200 0AFE 20 DB          BRA    EDF7
07210 0B00 81 34   EDF9   CMP A  £'4       Is it 4 ?
07220 0B02 26 05          BNE    *+7
07230 0B04 BD 0B1E        JSR    DECYCO    Yes, decrement y
07240 0B07 20 D2          BRA    EDF7
07250 0B09 BD 0B32        JSR    INCYCO    No, increment y
07260 0B0C 20 CD          BRA    EDF7
07270 0B0E 81 52   EDF10  CMP A  £'R       Is it R ?
07280 0B10 26 C9          BNE    EDF7      No, illegal-ignore
07290 0B12 7E 0A1F        JMP    EDFUNC    Yes,restart edit..
07300              *
07310 0B15 7D 1318 DECXCO TST    XCO      - DEC XCO WITHOUT
07320 0B18 27 03          BEQ    *+5      - ALLOWING XCO < 00
07330 0B1A 7A 1318        DEC    XCO
07340 0B1D 39             RTS
07350              *
07360 0B1E 7D 1319 DECYCO TST    YCO      - DEC YCO WITHOUT
07370 0B21 27 03          BEQ    *+5      - ALLOWING YCO < 00
07380 0B23 7A 1319        DEC    YCO
07390 0B26 39             RTS
07400              *
07410 0B27 F6 1318 INCXCO LDA B  XCO      - INC XCO WITHOUT
07420 0B2A C1 1F          CMP B  £$1F      - ALLOWING XCO > 1F
```

Assembly Listing of LPP Software Simulator

Part 1 - Control, Train/Test, Data Handler Sections

```
07430 0B2C 27 03          BEQ    *+5
07440 0B2E 7C 1318        INC    XCO
07450 0B31 39             RTS
07460                *
07470 0B32 F6 1319 INCYCO LDA B  YCO      - INC YCO WITHOUT
07480 0B35 C1 1F          CMP B  £$1F     - ALLOWING YCO > 1F
07490 0B37 27 03          BEQ    *+5
07500 0B39 7C 1319        INC    YCO
07510 0B3C 39             RTS
07520                *
07530 0B3D BD E0B1 NPASS  JSR    WAIT2    - GET NO. OF PASSES
07540 0B40 CE 04ED        LDX    £XNPASS - FROM TERMINAL
07550 0B43 BD E07E        JSR    PDATA1     "No.passes ? "
07560 0B46 BD E055        JSR    BYTE       Get 2 hex digits
07570 0B49 B7 030D        STA A  PASSES      into 'PASSES'
07580 0B4C 7E E0B1        JMP    WAIT2      Do CRLF
07590                *
07600                     END

TOTAL ERRORS 00000

ENTER PASS : 1P,2P,2L,2T
```

Assembly Listing of LPP Software Simulator

Part 2 - Memory Matrix Handler Section

```
00010                       NAM     LPPMM1
00020                       OPT     O,NOG,NOP
00030 3200                  ORG     $3200
00040                 *
00050                 * THIS PROGRAM REQUIRES THE FOLLOWING
00060                 * TO RUN CORRECTLY -
00070                 * 'LPPSR21'
00080                 * 'LPPF5'
00090                 *
00100       132B      ADDR    EQU     $132B   External Equates
00110       E1AC      INEEE   EQU     $E1AC   for Routines,
00120       E0B1      WAIT2   EQU     $E0B1       Variables,
00130       E0D0      MONTOR  EQU     $E0D0
00140       0306      FFEED   EQU     $0306
00150       0809      SMM     EQU     $0809   in: LPPF5
00160       0880      LMM     EQU     $0880       ($0100-0C00)
00170       0671      PREMM   EQU     $0671
00180       04E0      XREADY  EQU     $04E0       LPPSR21
00190       E07E      PDATA1  EQU     $E07E       ($1000-1900)
00200       E067      OUTHL   EQU     $E067
00210       E06B      OUTHR   EQU     $E06B       MSIBUG
00220       1332      PITO    EQU     $1332       ($E000-E400)
00230       1333      PIT1    EQU     $1333       System Monitor
00240       1334      PITX    EQU     $1334       in ROM
00250       1700      PREP    EQU     $1700
00260       132E      BITO    EQU     $132E
00270       132F      BIT1    EQU     $132F
00280       135F      BITX    EQU     $135F
00290       1713      GETTS   EQU     $1713
00300       4000      MMST    EQU     $4000
00310       1337      PRTON   EQU     $1337
00320       1338      PRTOFF  EQU     $1338
00330       E0CC      OUTS    EQU     $E0CC
00340       E1D1      OUTEEE  EQU     $E1D1
00350                 *
00360 3200 CE 3339    MMHAND  LDX     £XMM1       - START OF MEMORY
00370 3203 BD 1700            JSR     PREP        - MATRIX HANDLER
00380 3206 C1 43              CMP B   £'C         "MM Handler ? "
00390 3208 26 01              BNE     *+3         Is it C ?
00400 320A 39                 RTS                 Yes, go to control
00410 320B C1 41              CMP B   £'A         Is it A ?
00420 320D 26 03              BNE     *+5         Yes, print addrs.
00430 320F 7E 34D8            JMP     PADDR
00440 3212 C1 44              CMP B   £'D         Is it D ?
00450 3214 26 03              BNE     *+5         Yes, print MM data
00460 3216 7E 338C            JMP     PDATA
00470 3219 C1 54              CMP B   £'T         Is it T ?
00480 321B 26 03              BNE     *+5         Yes, totalize MM
00490 321D 7E 3585            JMP     TOTAL
00500 3220 C1 4B              CMP B   £'K         Is it K ? - Key in
00510 3222 26 DC              BNE     MMHAND      Illegal - ignore
00520                 *
00530 3224 86 11      KEYIN   LDA A   £$11        - KEYIN M.M.
00540 3226 BD 0682            JSR     PREMM+17    - MANUALLY FROM
00550 3229 CE 4000            LDX     £MMST       - TERMINAL
00560 322C FF 132B    CR1     STX     ADDR        Preset MM to $11s
00570 322F A6 00              LDA A   0,X
00580 3231 81 11              CMP A   £$11        Get next
00590 3233 26 30              BNE     CR6          MM cell = $11
00600 3235 BD 327F    CR5     JSR     PWIND       O/P window on
00610 3238 BD E1AC            JSR     INEEE        terminal,set
00620 323B B1 1332            CMP A   PITO         pixel reply
00630 323E 27 0F              BEQ     CR2         Is pixel '0' ?
00640 3240 B1 1333            CMP A   PIT1        Is pixel '1' ?
00650 3243 27 0F              BEQ     CR3
00660 3245 B1 1334            CMP A   PITX        Is pixel '?' ?
00670 3248 26 27              BNE     CR10        Illegal-reprompt
00680 324A B6 135F            LDA A   BITX        Get 'BITX'
00690 324D 20 08              BRA     CR4
00700 324F B6 132E    CR2     LDA A   BITO        Get 'BITO'
00710 3252 20 03              BRA     CR4
00720 3254 B6 132F    CR3     LDA A   BIT1        Get 'BIT1'
00730 3257 A7 00      CR4     STA A   0,X         Store in MM
00740 3259 BD 3577            JSR     CRLFSX      Do two CRLFs
00750 325C BD 3577            JSR     CRLFSX
00760 325F BD 3291            JSR     RANDR       Store in r+r cells
00770 3262 FE 132B            LDX     ADDR
00780 3265 08         CR6     INX                 Increment MM addr.
00790 3266 8C 4200            CPX     £MMST+$200
00800 3269 26 C1              BNE     CR1         End of MM ?
00810 326B BD 0809            JSR     SMM         Yes, store MM ?
00820 326E 7E 3200            JMP     MMHAND      Return
00830 3271 FF 3581    CR10    STX     SAVEX
00840 3274 CE 3395            LDX     £XWHAT      "Unrecognisable."
00850 3277 BD E07E            JSR     PDATA1
00860 327A FE 3581            LDX     SAVEX
00870 327D 20 B6              BRA     CR5         Re-prompt.....
00880                 *
00890 327F BD 3475    PWIND   JSR     P234        - PRINT WINDOW
00900 3282 BD 355C            JSR     PMMADD      Print :
00910 3285 BD 3577            JSR     CRLFSX       top line,
00920 3288 BD 348F            JSR     P105         MM address, CRLF,
00930 328B BD 3577            JSR     CRLFSX       centre line,CRLF,
00940 328E 7E 34AD            JMP     P876         bottom line....
00950                 *
00960 3291 FF 3583    RANDR   STX     SAVEX1  - ROTATE AND REFLECT
00970 3294 BD 32FE            JSR     ROT     - WINDOW AND STA A
00980 3297 BD 32BF            JSR     STAA    - IN ALL M.M. CELLS
00990 329A BD 32FE            JSR     ROT     - THUS GENERATED
01000 329D BD 32BF            JSR     STAA
01010 32A0 BD 32FE            JSR     ROT     Rotate window clock-
01020 32A3 BD 32BF            JSR     STAA     wise to 4 places,
01030 32A6 BD 3313            JSR     VREF
01040 32A9 BD 32BF            JSR     STAA    Reflect vertically,
01050 32AC BD 32FE            JSR     ROT
01060 32AF BD 32BF            JSR     STAA    Rotate again to
01070 32B2 BD 32FE            JSR     ROT      4 places,
01080 32B5 BD 32BF            JSR     STAA
```

Assembly Listing of LPP Software Simulator

Part 2 - Memory Matrix Handler Section

```
01090 32B8 BD 32FE        JSR    ROT     Store acc.in 8xMM
01100 32BB BD 32BF        JSR    STAA     addrs.thus made
01110 32BE 39             RTS
01120              *
01130 32BF FE 132B STAA   LDX    ADDR      - STORE C(ADDR)
01140 32C2 A6 00          LDA A  0,X       - IN LOC. POINTED
01150 32C4 FE 3583        LDX    SAVEX1    - TO BY SAVEX1
01160 32C7 A7 00          STA A  0,X
01170 32C9 39             RTS
01180              *
01190 32CA 0C      FSH    CLC              - FIRST SHIFT OF
01200 32CB B6 3584        LDA A  SAVEX1+1 - M.M. ADDRESS
01210 32CE 46             ROR A
01220 32CF B7 3584        STA A  SAVEX1+1   Shift eight
01230 32D2 B6 3583        LDA A  SAVEX1      neighbour
01240 32D5 46             ROR A              pixels (not P0)
01250 32D6 B7 3583        STA A  SAVEX1      to a single
01260 32D9 24 08          BCC    *+10        byte to
01270 32DB B6 3584        LDA A  SAVEX1+1    facilitate
01280 32DE 8A 80          ORA A  £$80        'ROT+VREF'
01290 32E0 B7 3584        STA A  SAVEX1+1
01300 32E3 39             RTS
01310              *
01320 32E4 0C      LSH    CLC              - LAST SHIFT OF
01330 32E5 B6 3583        LDA A  SAVEX1    - M.M. ADDRESS
01340 32E8 49             ROL A
01350 32E9 B7 3583        STA A  SAVEX1
01360 32EC B6 3584        LDA A  SAVEX1+1  Re-format eight
01370 32EF 49             ROL A              neighbour pixels
01380 32F0 B7 3584        STA A  SAVEX1+1    from a single
01390 32F3 24 08          BCC    *+10        byte to two
01400 32F5 B6 3583        LDA A  SAVEX1      bytes
01410 32F8 8A 01          ORA A  £$01
01420 32FA B7 3583        STA A  SAVEX1
01430 32FD 39             RTS
01440              *
01450 32FE BD 32CA ROT    JSR    FSH      - ROTATE WINDOW
01460 3301 B6 3584        LDA A  SAVEX1+1 - 90 DEGREES
01470 3304 49             ROL A            - CLOCKWISE
01480 3305 49             ROL A
01490 3306 16             TAB              Re-arrange byte
01500 3307 C4 FC          AND B  £$FC       in 'SAVEX1+1'
01510 3309 49             ROL A             to simulate
01520 330A 84 03          AND A  £$03       rotation
01530 330C 1B             ABA
01540 330D B7 3584        STA A  SAVEX1+1
01550 3310 7E 32E4        JMP    LSH
01560              *
01570 3313 BD 32CA VREF   JSR    FSH      - REFLECT WINDOW
01580 3316 5F             CLR B            - ABOUT VERTICAL
01590 3317 B6 3584        LDA A  SAVEX1+1 - AXIS
01600 331A 49             ROL A
01610 331B 56             ROR B
01620 331C 49             ROL A
01630 331D 56             ROR B            Re-arrange byte
01640 331E 49             ROL A             in 'SAVEX1+1'
01650 331F 56             ROR B             to simulate
01660 3320 F7 3584        STA B  SAVEX1+1   reflection
01670 3323 49             ROL A
01680 3324 56             ROR B
01690 3325 49             ROL A
01700 3326 56             ROR B
01710 3327 49             ROL A
01720 3328 56             ROR B
01730 3329 49             ROL A
01740 332A 56             ROR B
01750 332B 49             ROL A
01760 332C 56             ROR B
01770 332D 54             LSR B
01780 332E 54             LSR B
01790 332F 54             LSR B
01800 3330 FA 3584        ORA B  SAVEX1+1
01810 3333 F7 3584        STA B  SAVEX1+1
01820 3336 7E 32E4        JMP    LSH
01830              *                    Text Strings :

01840 3339 4D      XMM1   FCC    'MEMORY MATRIX HANDLER - '
01850 3351 0D0A           FDB    $0D0A
01860 3353 50             FCC    'PRINT ADDRESSES ,
                                  DATA SETS , KEYIN , '
01870 337E 54             FCC    'TOTALIZE OR CONTROL ? '
01880 3394 04             FCB    4
01890 3395 20      XWHAT  FCC    ' UNRECOGNISABLE -
                                  USE  " , / ? / O " '
01900 33B8 0D0A           FDB    $0D0A
01910 33BA 0A04           FDB    $0A04
01920              *
01930 33BC BD 088E PDATA  JSR    LMM+14    - LOAD AND PRINT
01940 33BF D1 4E          CMP B  $'N       - M.M. DATA SET
01950 33C1 26 03          BNE    DMM2      "Load MM ? "
01960 33C3 7E 3200        JMP    MMHAND
01970 33C6 CE 04E0 DMM2   LDX    £XREADY   "Ready ? "
01980 33C9 BD 1700        JSR    PREP
01990 33CC B6 1337        LDA A  PRTON     Turn printer ON
02000 33CF BD E1D1        JSR    OUTEEE
02010 33D2 CE 4000        LDX    £MMST     Get 1st MM addr.
02020 33D5 BD 3577 DMM10  JSR    CRLFSX
02030 33D8 BD 3577        JSR    CRLFSX
02040 33DB BD 3577        JSR    CRLFSX    Do 6x CRLFs
02050 33DE BD 3577        JSR    CRLFSX
02060 33E1 BD 3577        JSR    CRLFSX
02070 33E4 BD 3577        JSR    CRLFSX
02080 33E7 5F      DMM6   CLR B            Clear column count
02090 33E8 BD 3577        JSR    CRLFSX    Do CRLF
02100 33EB BD 355C        JSR    PMMADD    Print MM address
02110 33EE 08             INX              Incr. MM addr.
02120 33EF 5C             INC B            Incr. col.count
02130 33F0 C1 10          CMP B  £$10      Col.count = 16 ?
```

Assembly Listing of LPP Software Simulator

Part 2 - Memory Matrix Handler Section

```
02140 33F2 26 F7          BNE    *-7        No, loop back
02150 33F4 BD 3464        JSR    DEX16      Yes, decrement
02160 33F7 BD 3577        JSR    CRLFSX      MM addr. by 16
02170 33FA 5F             CLR B             Clear column count
02180 33FB BD E0CC DMM8   JSR    OUTS       Print space
02190 33FE A6 00          LDA A  0,X        Get MM cell data
02200 3400 B1 132E        CMP A  BIT0       Data = 'BIT0' ?
02210 3403 27 0F          BEQ    DMM3
02220 3405 B1 132F        CMP A  BIT1       Data = 'BIT1' ?
02230 3408 27 05          BEQ    DMM4
02240 340A B6 1334        LDA A  PITX       Get 'PITX'
02250 340D 20 08          BRA    DMM5
02260 340F B6 1333 DMM4   LDA A  PIT1       Get 'PIT1'
02270 3412 20 03          BRA    DMM5
02280 3414 B6 1332 DMM3   LDA A  PIT0       Get 'PIT0'
02290 3417 BD E1D1 DMM5   JSR    OUTEEE     Print 'PIT' char.
02300 341A BD 346C        JSR    OUT3S      Print 3 spaces
02310 341D 08             INX               Incr. MM addr.
02320 341E 5C             INC B             Incr. column count
02330 341F C1 10          CMP B  £$10       Col.count = 16 ?
02340 3421 26 D8          BNE    DMM8           No, loop back
02350 3423 BD 3577        JSR    CRLFSX         Yes, do CRLF
02360 3426 8C 4080        CPX    £MMST+$80      1st quarter
02370 3429 27 1E          BEQ    DMM7               done ?
02380 342B 8C 4100        CPX    £MMST+$100     2nd quarter
02390 342E 27 2B          BEQ    DMM9               done ?
02400 3430 8C 4180        CPX    £MMST+$180     3rd quarter
02410 3433 27 14          BEQ    DMM7               done ?
02420 3435 8C 4200        CPX    £MMST+$200     4th quarter
02430 3438 26 AD          BNE    DMM6               done ?
02440 343A B6 0306        LDA A  FFEED      End, do formfeed
02450 343D BD E1D1        JSR    OUTEEE
02460 3440 B6 1338        LDA A  PRTOFF     Turn printer OFF
02470 3443 BD E1D1        JSR    OUTEEE
02480 3446 7E 3200        JMP    MMHAND     Return
02490 3449 BD 3577 DMM7   JSR    CRLFSX
02500 344C BD 3577        JSR    CRLFSX
02510 344F BD 3577        JSR    CRLFSX     Do 5x CRLFs
02520 3452 BD 3577        JSR    CRLFSX
02530 3455 BD 3577        JSR    CRLFSX
02540 3458 7E 33E7        JMP    DMM6
02550 345B B6 0306 DMM9   LDA A  FFEED      Do formfeed
02560 345E BD E1D1        JSR    OUTEEE
02570 3461 7E 33D5        JMP    DMM10
02580                *
02590 3464 5F       DEX16 CLR B             - 'DEX' BY 16
02600 3465 09             DEX
02610 3466 5C             INC B
02620 3467 C1 10          CMP B  £$10       Finished ?
02630 3469 26 FA          BNE    *-4
02640 346B 39             RTS
02650                *
02660 346C BD E0CC OUT3S  JSR    OUTS       - PRINT 3 SPACES
02670 346F BD E0CC        JSR    OUTS
02680 3472 7E E0CC        JMP    OUTS
02690                *
02700 3475 FF 3581 P234   STX    SAVEX      - PRINT ADDR BITS
02710 3478 B6 3582        LDA A  SAVEX+1    - P2,P3,P4 AND
02720 347B 46             ROR A             - 2 SPACES
02730 347C 46             ROR A
02740 347D 46             ROR A             Get P2 into carry
02750 347E BD 34C8        JSR    PCARRY     Print carry
02760 3481 46             ROR A             Get P3 into carry
02770 3482 BD 34C8        JSR    PCARRY     Print carry
02780 3485 46             ROR A             Get P4 into carry
02790 3486 BD 34C8        JSR    PCARRY     Print carry
02800 3489 BD E0CC        JSR    OUTS
02810 348C 7E E0CC        JMP    OUTS       Do 2 spaces
02820                *
02830 348F FF 3581 P105   STX    SAVEX      - PRINT ADDR BITS
02840 3492 B6 3582        LDA A  SAVEX+1    - P1,P0,P5 AND
02850 3495 46             ROR A             - 2 SPACES
02860 3496 46             ROR A             Get P1 into carry
02870 3497 BD 34C8        JSR    PCARRY     Print carry
02880 349A 49             ROL A             Get P0 into carry
02890 349B BD 34C8        JSR    PCARRY     Print carry
02900 349E B6 3582        LDA A  SAVEX+1
02910 34A1 49             ROL A
02920 34A2 49             ROL A
02930 34A3 49             ROL A             Get P5 into carry
02940 34A4 BD 34C8        JSR    PCARRY     Print carry
02950 34A7 BD E0CC        JSR    OUTS
02960 34AA 7E E0CC        JMP    OUTS       Do 2 spaces
02970                *
02980 34AD FF 3581 P876   STX    SAVEX      - PRINT ADDR BITS
02990 34B0 B6 3581        LDA A  SAVEX      - P8,P7,P6 AND
03000 34B3 46             ROR A             - 2 SPACES
03010 34B4 BD 34C8        JSR    PCARRY     Get P8, print
03020 34B7 B6 3582        LDA A  SAVEX+1
03030 34BA 49             ROL A             Get P7 into carry
03040 34BB BD 34C8        JSR    PCARRY     Print carry
03050 34BE 49             ROL A             Get P6 into carry
03060 34BF BD 34C8        JSR    PCARRY     Print carry
03070 34C2 BD E0CC        JSR    OUTS
03080 34C5 7E E0CC        JMP    OUTS       Do 2 spaces
03090                *
03100 34C8 36       PCARRY PSH A            - PRINT '1' OR '0'
03110 34C9 24 05          BCC    *+7        - DEPENDANT ON
03120 34CB B6 1333        LDA A  PIT1       - CARRY BIT
03130 34CE 20 03          BRA    *+5
03140 34D0 B6 1332        LDA A  PIT0       Get 'PIT0 or 1'
03150 34D3 BD E1D1        JSR    OUTEEE     Print
03160 34D6 32             PUL A
03170 34D7 39             RTS
03180                *
03190 34D8 CE 04E0 PADDR  LDX    £XREADY    - PRINT ADDRESS
03200 34DB BD 1700        JSR    PREP       - OF M.M. WITH
03210 34DE CE 4000        LDX    £MMST      - FEATURES
```

Assembly Listing of LPP Software Simulator

Part 2 - Memory Matrix Handler Section

```
03220 34E1 B6 1337        LDA A  PRTON     Turn printer ON
03230 34E4 BD E1D1        JSR    OUTEEE
03240 34E7 5F       AD3   CLR B            Clear column count
03250 34E8 BD 3577        JSR    CRLFSX    Do 2x CRLFs
03260 34EB BD 3577        JSR    CRLFSX
03270 34EE BD 355C        JSR    PMMADD    Print MM address
03280 34F1 08             INX              Incr.MM address
03290 34F2 5C             INC B            Incr.col.count
03300 34F3 C1 10          CMP B  £$10     Col.count = 16 ?
03310 34F5 26 F7          BNE    *-7       No, loop back....
03320 34F7 BD 3464        JSR    DEX16     Decrement MM addr.
03330 34FA BD 3577        JSR    CRLFSX
03340 34FD BD 3577        JSR    CRLFSX    Do 2x CRLFs
03350 3500 5F             CLR B            Clear column count
03360 3501 BD 34/5        JSR    P234      Print top line
03370 3504 08             INX              Incr.MM addr.
03380 3505 5C             INC B            Incr.col.count
03390 3506 C1 10          CMP B  £$10     Col.count = 16 ?
03400 3508 26 F7          BNE    *-7       No, loop back....
03410 350A BD 3464        JSR    DEX16     Decrement MM addr.
03420 350D BD 3577        JSR    CRLFSX    Do CRLF
03430 3510 5F             CLR B            Clear column count
03440 3511 BD 348F        JSR    P105      Print centre line
03450 3514 08             INX              Incr.MM addr.
03460 3515 5C             INC B            Incr.col.count
03470 3516 C1 10          CMP B  £$10     Col.count = 16 ?
03480 3518 26 F7          BNE    *-7       No, loop back....
03490 351A BD 3464        JSR    DEX16     Decrment MM addr.
03500 351D BD 3577        JSR    CRLFSX    Do CRLF
03510 3520 5F             CLR B            Clear column count
03520 3521 BD 34AD        JSR    P876      Print bottom line
03530 3524 08             INX              Incr.MM addr.
03540 3525 5C             INC B            Incr.col.count
03550 3526 C1 10          CMP B  £$10     Col.count = 16 ?
03560 3528 26 F7          BNE    *-7       No, loop back....
03570 352A BD 3577        JSR    CRLFSX
03580 352D BD 3577        JSR    CRLFSX      Do 2x CRLFs
03590 3530 8C 4080        CPX    £MMST+$80   1st quarter
03600 3533 27 1F          BEQ    AD2             done ?
03610 3535 8C 4100        CPX    £MMST+$100 2nd quarter
03620 3538 27 1A          BEQ    AD2             done
03630 353A 8C 4180        CPX    £MMST+$180 3rd quarter
03640 353D 27 15          BEQ    AD2             done ?
03650 353F 8C 4200        CPX    £MMST+$200 4th quarter
03660 3542 26 A3          BNE    AD3             done ?
03670 3544 B6 0306        LDA A  FFEED     Do formfeed
03680 3547 BD E1D1        JSR    OUTEEE
03690 354A B6 1338        LDA A  PRTOFF    Turn printer OFF
03700 354D BD E1D1        JSR    OUTEEE
03710 3550 7E 3200        JMP    MMHAND    Return
03720 3553 39       AD1   RTS
03730 3554 B6 0306  AD2   LDA A  FFEED     Do formfeed
03740 3557 BD E1D1        JSR    OUTEEE
03750 355A 20 8B          BRA    AD3
03760                *
03770 355C FF 3581 PMMADD STX    SAVEX     - PRINT M.M. ADDR
03780 355F B6 3581        LDA A  SAVEX     - IN 3 DIGIT HEX
03790 3562 BD E06B        JSR    OUTHR     - AND 2 SPACES
03800 3565 B6 3582        LDA A  SAVEX+1   Print:
03810 3568 BD E067        JSR    OUTHL      RH digit(SAVEX)
03820 356B B6 3582        LDA A  SAVEX+1    LH digit(SAVEX+1)
03830 356E BD E06B        JSR    OUTHR      RH digit(SAVEX+1)
03840 3571 BD E0CC        JSR    OUTS
03850 3574 7E E0CC        JMP    OUTS      Do 2 spaces
03860                *
03870 3577 FF 3581 CRLFSX STX    SAVEX     - 'CRLF' WHILST
03880 357A BD E0B1        JSR    WAIT2     - SAVING 'X'
03890 357D FE 3581        LDX    SAVEX
03900 3580 39             RTS
03910                *
03920 3581 0002    SAVEX  RMB    2         Temporary
03930 3583 0002    SAVEX1 RMB    2         Variable stores
03940                *
03950 3585 BD 088E TOTAL  JSR    LMM+14    - TOTALIZE BITS SET
03960 3588 B6 132E        LDA A  BIT0      - TO BIT 0,X,1
03970 358B BD 35A0        JSR    CNT       "Load MM ? "
03980 358E B6 135F        LDA A  BITX      Count 'BIT0's,
03990 3591 BD 35A0        JSR    CNT       Count 'BITX's,
04000 3594 B6 132F        LDA A  BIT1      Count 'BIT1's.
04010 3597 BD 35A0        JSR    CNT
04020 359A BD E0B1        JSR    WAIT2     Do CRLF
04030 359D 7E 3200        JMP    MMHAND
04040                *
04050 35A0 36      CNT    PSH A            - PRINT LABEL AND
04060 35A1 36             PSH A            - TOTAL
04070 35A2 CE 35F1        LDX    £XNOLOC   "No.of MM locs="
04080 35A5 BD E07E        JSR    PDATA1
04090 35A8 32             PUL A
04100 35A9 B1 132E        CMP A  BIT0      Counting 'BIT0's ?
04110 35AC 26 04          BNE    CNT1
04120 35AE 86 30          LDA A  £'0       Yes, print "0"
04130 35B0 20 0B          BRA    CNT3
04140 35B2 B1 135F CNT1   CMP A  BITX      Counting 'BITX's ?
04150 35B5 26 04          BNE    CNT2
04160 35B7 86 58          LDA A  £'X       Yes, print "X"
04170 35B9 20 02          BRA    CNT3
04180 35BB 86 31   CNT2   LDA A  £'1       No,  print "1"
04190 35BD BD E1D1 CNT3   JSR    OUTEEE
04200 35C0 CE 360F        LDX    £XDASH    Print dash " - "
04210 35C3 BD E07E        JSR    PDATA1
04220 35C6 7F 3581        CLR    SAVEX     Clear 16-bit count
04230 35C9 7F 3582        CLR    SAVEX+1
04240 35CC CE 4000        LDX    £MMST     Get 1st MM addr.
04250 35CF 32             PUL A
04260 35D0 A1 00   CNT4   CMP A  0,X       C(cell)=A ?
04270 35D2 26 03          BNE    *+5
04280 35D4 BD 35E3        JSR    INCSX     Yes, incr. count
04290 35D7 08             INX                Incr. MM addr.
```

Assembly Listing of LPP Software Simulator

Part 2 - Memory Matrix Handler Section

```
04300 35D8 8C 4200          CPX    £MMST+$200 End of MM ?
04310 35DB 26 F3            BNE    CNT4       No, loop back...
04320 35DD BD 355F          JSR    PMMADD+3   Print result,
04330 35E0 7E E0B1          JMP    WAIT2      Do CRLF
04340                 *
04350 35E3 FF 3583 INCSX    STX    SAVEX1  - INC SAVEX,
04360 35E6 FE 3581          LDX    SAVEX   - WHILST SAVING 'X'
04370 35E9 08               INX
04380 35EA FF 3581          STX    SAVEX    Use temp. store
04390 35ED FE 3583          LDX    SAVEX1   of 'SAVEX1' for'X'
04400 35F0 39               RTS
04410                 *                     Text Strings :

04420 35F1 4E       XNOLOC  FCC    'NO. OF M.M. LOCS.
                                    SET TO BIT '
04430 360E 04               FCB    4
04440 360F 20       XDASH   FCC    '  -  '
04450 3614 04               FCB    4
04460                 *
04470                       END

TOTAL ERRORS 00000

ENTER PASS : 1P,2P,2L,2T
```

Assembly Listing of LPP Software Simulator

Part 3 - Sub-routine Facilities Section
(Used by Parts 1 and 2)

```
00010                          NAM    LPPSR21
00020                          OPT    O,NOG,NOP
00030                  *
00040        B000      VDUST  EQU    $B000   External Equates
00050        3E00      BUFST  EQU    $3E00   for Routines,
00060        A960      LEADER EQU    $A960       Variables,
00070        A990      PUNCH  EQU    $A990
00080        0000      TRACK  EQU    $0000   in: MSIBUG
00090        E0B1      WAIT2  EQU    $E0B1       ($E000-E400)
00100        E1D1      OUTEEE EQU    $E1D1       System Monitor
00110        E0CC      OUTS   EQU    $E0CC       in ROM
00120        0001      SECTOR EQU    $0001
00130        000C      NOSECT EQU    $000C       DOS
00140        000A      EMEMH  EQU    $000A       ($2400-3200)
00150        2439      READ   EQU    $2439       Disk Operating
00160        2758      WRITE  EQU    $2758       System in RAM
00170        001B      ERRH   EQU    $001B
00180        A880      IPTAPE EQU    $A880       LIB1
00190        A980      IPTIL  EQU    $A980       ($A800-AA00)
00200        E07E      PDATA1 EQU    $E07E       Set of Library
00210        E0B1      WAIT2  EQU    $E0B1       Routines -
00220        E1AC      INEEE  EQU    $E1AC
00230        E089      INHEX  EQU    $E089
00240        E047      BADDR  EQU    $E047
00250        E055      BYTE   EQU    $E055
00260                  *
00270 1000                     ORG    $1000
00280 1000 FF 1312 INX64       STX    DSUM1     - 'INX' BY 64
00290 1003 CE 0040             LDX    £$0040      'DSUM1' = X,
00300 1006 FF 1314             STX    DSUM2       'DSUM2' = $0040,
00310 1009 BD 1018             JSR    DSUM        Add...
00320 100C FE 1316             LDX    DSUM3       X = 'DSUM3'
00330 100F 39          DUMMY   RTS
00340                  *
00350 1010 FF 1312 DEX64       STX    DSUM1     - 'DEX' BY 64
00360 1013 CE FFC0             LDX    £$FFC0      'DSUM2' = $FFC0,
00370 1016 20 EE               BRA    INX64+6     Go to INX64...
00380                  *
00390 1018 B6 1313 DSUM        LDA A  DSUM1+1   - DOUBLE BYTE
00400 101B F6 1312             LDA B  DSUM1     - ADDITION
00410 101E BB 1315             ADD A  DSUM2+1     Add LS bytes,
00420 1021 F9 1314             ADC B  DSUM2       Add MS bytes
00430 1024 B7 1317             STA A  DSUM3+1      with carry,
00440 1027 F7 1316             STA B  DSUM3       Store result.
00450 102A 39                  RTS
00460                  *
00470 102B FE 130A BSET        LDX    BUFOST    - SET BUFLOC
00480 102E FF 1310             STX    BUFLOC    - TO BUFOST
00490 1031 39                  RTS
00500                  *
00510 1032 FE 1302 VSET        LDX    VDUOST    - SET VDULOC
00520 1035 B6 1300             LDA A  VDUNO     - TO VDU START
00530 1038 27 03               BEQ    *+5         'VDUNO'=0 or 1 ?
00540 103A BD 10AD             JSR    INX32       1, so shift
00550 103D FF 130E             STX    VDULOC      'VDULOC'32 places
00560 1040 39                  RTS                 to the right
00570                  *
00580 1041 7F 000C B2D         CLR    NOSECT    - BUF () DISK
00590 1044 7C 000C             INC    NOSECT      Set no. sectors
00600 1047 CE FFFF             LDX    £$FFFF       equal to ONE
00610 104A DF 0A               STX    EMEMH       Initialize DOS
00620 104C BD 102B             JSR    BSET        Set 'X'=Buffer
00630 104F B6 131C             LDA A  B2DDIR              addr.
00640 1052 27 03               BEQ    *+5         Direction ?
00650 1054 7E 2439             JMP    READ        D to B, so read,
00660 1057 7E 2758             JMP    WRITE       B to D, so write.
00670                  *
00680 105A BD 1032 B2V         JSR    VSET      - BUF ( ) VDU
00690 105D BD 102B             JSR    BSET        Set 'B/V' pointers
00700 1060 7F 1318             CLR    XCO         Clear x
00710 1063 7F 1319 B2V2        CLR    YCO         Clear y
00720 1066 7F 131A B2V3        CLR    ZCO         Clear z (bit no.)
00730 1069 B6 131B B2V1        LDA A  B2VDIR      Direction ?
00740 106C 27 05               BEQ    *+7
00750 106E BD 10B6             JSR    VTOB        V to B.....
00760 1071 20 03               BRA    *+5
00770 1073 BD 10F6             JSR    BTOV        B to V.....
00780 1076 FE 130E             LDX    VDULOC
00790 1079 08                  INX                Increment VDU
00800 107A 01                  NOP                 pointer
00810 107B FF 130E             STX    VDULOC
00820 107E 7C 1319             INC    YCO         Increment y,z
00830 1081 7C 131A             INC    ZCO
00840 1084 B6 131A             LDA A  ZCO
00850 1087 81 08               CMP A  £$08        z = 8 ?
00860 1089 2D DE               BLT    B2V1        No, loop back....
00870 108B FE 1310             LDX    BUFLOC
00880 108E 08                  INX                Yes, increment
00890 108F FF 1310             STX    BUFLOC       buffer pointer
00900 1092 B6 1319             LDA A  YCO
00910 1095 81 20               CMP A  £$20        y = 32 ?
00920 1097 2D CD               BLT    B2V3        No, loop back....
00930 1099 7C 1318             INC    XCO         Yes, increment x
00940 109C FE 130E             LDX    VDULOC
00950 109F BD 10AD             JSR    INX32       Increment VDU
00960 10A2 FF 130E             STX    VDULOC       pointer by 32
00970 10A5 B6 1318             LDA A  XCO
00980 10A8 81 20               CMP A  £$20        x = 32 ?
00990 10AA 2D B7               BLT    B2V2        No, loop back...
01000 10AC 39                  RTS                Yes, return.....
01010                  *
01020 10AD FF 1312 INX32       STX    DSUM1     - 'INX' BY 32
01030 10B0 CE 0020             LDX    £$0020      'DSUM2' = $0020,
01040 10B3 7E 1006             JMP    INX64+6     Go to 'INX64'...
01050                  *
01060 10B6 86 80   VTOB        LDA A  £$80      - TRANSFER BYTE
01070 10B8 F6 131A             LDA B  ZCO       - VDU > BIT BUF
01080 10BB 27 04               BEQ    *+6         Set up bit mask in
```

Assembly Listing of LPP Software Simulator

Part 3 - Sub-routine Facilities Section
(Used by Parts 1 and 2)

```
01090 10BD 44              LSR A              acc. at position
01100 10BE 5A              DEC B              given by z
01110 10BF 20 FA           BRA     *-4
01120 10C1 FE 130E         LDX     VDULOC     Get byte (pixel)
01130 10C4 E6 00           LDA B   0,X        from VDU
01140 10C6 37              PSH B              Stack pixel
01150 10C7 36              PSH A
01160 10C8 F1 132F         CMP B   BIT1       Is pixel = "1" ?
01170 10CB 26 07           BNE     *+9
01180 10CD FE 1310         LDX     BUFLOC     Yes, OR buffer
01190 10D0 AA 00           ORA A   0,X        byte with bit
01200 10D2 20 06           BRA     *+8        mask
01210 10D4 43              COM A              No, invert mask
01220 10D5 FE 1310         LDX     BUFLOC     and AND buffer
01230 10D8 A4 00           AND A   0,X        byte with mask
01240 10DA A7 00           STA A   0,X
01250 10DC FE 1310         LDX     BUFLOC     Upper half of
01260 10DF BD 1000         JSR     INX64      buffer to store
01270 10E2 BD 1000         JSR     INX64      tri-state pixels
01280 10E5 32              PUL A
01290 10E6 33              PUL B
01300 10E7 F1 132E         CMP B   BIT0       Is pixel = "0" ?
01310 10EA 27 04           BEQ     *+6
01320 10EC AA 00           ORA A   0,X        No,OR buf.byte
01330 10EE 20 03           BRA     *+5        with mask
01340 10F0 43              COM A              Invert mask,AND
01350 10F1 A4 00           AND A   0,X        with buf.byte
01360 10F3 A7 00           STA A   0,X
01370 10F5 39              RTS
01380              *
01390 10F6 FE 1310 BTOV    LDX     BUFLOC     - TRANSFER BIT
01400 10F9 A6 00           LDA A   0,X        - BUF ) BYTE VDU
01410 10FB F6 131A         LDA B   ZCO        Get buffer byte
01420 10FE 27 04           BEQ     *+6
01430 1100 49              ROL A              Rotate byte so
01440 1101 5A              DEC B              carry contains
01450 1102 20 FA           BRA     *-4        bit pointed to
01460 1104 49              ROL A              by 'ZCO'
01470 1105 25 1E           BCS     BTOV2      Is bit = "1" ?
01480 1107 BD 1000         JSR     INX64      No, so test upper
01490 110A BD 1000         JSR     INX64      half of buffer
01500 110D A6 00           LDA A   0,X        Get upper buf.byte
01510 110F F6 131A         LDA B   ZCO
01520 1112 27 04           BEQ     *+6        Rotate byte so
01530 1114 49              ROL A              carry has bit
01540 1115 5A              DEC B              pointed to by
01550 1116 20 FA           BRA     *-4        'ZCO'
01560 1118 49              ROL A
01570 1119 25 05           BCS     *+7        Is bit = "1"·?
01580 111B B6 132E         LDA A   BIT0       No, set 'BIT0',
01590 111E 20 08           BRA     BTOV1
01600 1120 B6 135F         LDA A   BITX       Yes,set 'BITX',
01610 1123 20 03           BRA     BTOV1
01620 1125 B6 132F BTOV2   LDA A   BIT1       Get 'BIT1',
01630 1128 FE 130E BTOV1   LDX     VDULOC
01640 112B A7 00           STA A   0,X        Store pixel on VDU
01650 112D 39              RTS
01660              *
01670 112E BD 1032 XYLOC   JSR     VSET       - CALC. VDULOC
01680 1131 7F 1314         CLR     DSUM2      - FROM X,Y,VDUNO
01690 1134 FF 1312         STX     DSUM1      Set top left
01700 1137 B6 1319         LDA A   YCO        VDU addr.
01710 113A 01              NOP                Add 'YCO'
01720 113B B7 1315         STA A   DSUM2+1
01730 113E BD 1018         JSR     DSUM
01740 1141 FE 1316         LDX     DSUM3
01750 1144 FF 1312         STX     DSUM1
01760 1147 B6 1318         LDA A   XCO        Get x
01770 114A 44              LSR A
01780 114B 44              LSR A              Shift, to
01790 114C B7 1314         STA A   DSUM2      multiply by 64
01800 114F B6 1318         LDA A   XCO
01810 1152 46              ROR A
01820 1153 46              ROR A
01830 1154 46              ROR A
01840 1155 84 C0           AND A   £$C0
01850 1157 B7 1315         STA A   DSUM2+1
01860 115A BD 1018         JSR     DSUM       Add to result
01870 115D FE 1316         LDX     DSUM3
01880 1160 FF 130E         STX     VDULOC     Put in 'VDULOC'
01890 1163 39              RTS
01900              *
01910 1164 CE 0000 EDGET   LDX     £$0000     - TEST FOR EDGE
01920 1167 FF 1327         STX     LHE        - POINT
01930 116A FF 1329         STX     RHE        Clear top,l,r,
01940 116D B6 1319         LDA A   YCO        bottom flags
01950 1170 26 05           BNE     ET1        Is y = 0 ?
01960 1172 73 1327         COM     LHE        Yes, set l.flag
01970 1175 20 07           BRA     ET2
01980 1177 81 1F   ET1     CMP A   £$1F       Is y = 31 ?
01990 1179 26 03           BNE     ET2
02000 117B 73 1329         COM     RHE        Yes, set r.flag
02010 117E B6 1318 ET2     LDA A   XCO        Is x = 0 ?
02020 1181 26 05           BNE     ET3
02030 1183 73 1328         COM     THE        Yes, set top flag
02040 1186 20 07           BRA     ET4
02050 1188 81 1F   ET3     CMP A   £$1F       Is x = 31 ?
02060 118A 26 03           BNE     ET4        Yes,set bot.flag
02070 118C 73 132A         COM     BHE
02080 118F 39      ET4     RTS
02090              *
02100 1190 BD 112E GETPO   JSR     XYLOC      - GET PO FROM VDU
02110 1193 A6 00           LDA A   0,X        Get VDU address,
02120 1195 B7 131E         STA A   PO         Put pixel in 'PO'
02130 1198 39              RTS
02140              *
02150 1199 BD 112E PUTPO   JSR     XYLOC      - PUT PO ONTO VDU
02160 119C B6 131E         LDA A   PO         Get VDU address,
```

Assembly Listing of LPP Software Simulator

Part 3 - Sub-routine Facilities Section
(Used by Parts 1 and 2)

```
02170 119F A7 00          STA A  0,X       Put pixel on VDU
02180 11A1 39             RTS
02190                *
02200 11A2 BD 1164 GETWIN JSR    EDGET     - GET P0-P8
02210 11A5 BD 112E        JSR    XYLOC     - FROM VDU
02220 11A8 BD 1010        JSR    DEX64     Test for edges,
02230 11AB 09             DEX              Get VDU address,
02240 11AC 01             NOP              Offset....
02250 11AD A6 41          LDA A  $41,X     Get P0
02260 11AF B7 131E        STA A  P0
02270 11B2 7D 1327        TST    LHE       L.flag = 1 ?
02280 11B5 26 04          BNE    GWP1
02290 11B7 A6 40          LDA A  $40,X     No, set P1
02300 11B9 20 03          BRA    GWP1+3
02310 11BB B6 131D GWP1   LDA A  EDGE      Yes, set edge
02320 11BE B7 131F        STA A  P1        Store P1
02330 11C1 7D 1327        TST    LHE       L.flag = 1 ?
02340 11C4 26 09          BNE    GWP2
02350 11C6 7D 1328        TST    THE       Top flag = 1 ?
02360 11C9 26 04          BNE    GWP2
02370 11CB A6 00          LDA A  $0,X      No, set P2
02380 11CD 20 03          BRA    GWP2+3
02390 11CF B6 131D GWP2   LDA A  EDGE      Yes, set edge
02400 11D2 B7 1320        STA A  P2        Store P2
02410 11D5 7D 1328        TST    THE       Top flag = 1 ?
02420 11D8 26 04          BNE    GWP3
02430 11DA A6 01          LDA A  $1,X      No, set P3
02440 11DC 20 03          BRA    GWP3+3
02450 11DE B6 131D GWP3   LDA A  EDGE      Yes, set edge
02460 11E1 B7 1321        STA A  P3        Store P3
02470 11E4 7D 1328        TST    THE       Top flag = 1 ?
02480 11E7 26 09          BNE    GWP4
02490 11E9 7D 1329        TST    RHE       R.flag = 1 ?
02500 11EC 26 04          BNE    GWP4
02510 11EE A6 02          LDA A  $2,X      No, set P4
02520 11F0 20 03          BRA    GWP4+3
02530 11F2 B6 131D GWP4   LDA A  EDGE      Yes, set edge
02540 11F5 B7 1322        STA A  P4        Store P4
02550 11F8 7D 1329        TST    RHE       R.flag = 1 ?
02560 11FB 26 04          BNE    GWP5
02570 11FD A6 42          LDA A  $42,X     No, set P5
02580 11FF 20 03          BRA    GWP5+3
02590 1201 B6 131D GWP5   LDA A  EDGE      Yes, set edge
02600 1204 B7 1323        STA A  P5        Store P5
02610 1207 7D 1329        TST    RHE       R.flag = 1 ?
02620 120A 26 09          BNE    GWP6
02630 120C 7D 132A        TST    BHE       Bot.flag = 1 ?
02640 120F 26 04          BNE    GWP6
02650 1211 A6 82          LDA A  $82,X     No, set P6
02660 1213 20 03          BRA    GWP6+3
02670 1215 B6 131D GWP6   LDA A  EDGE      Yes, set edge
02680 1218 B7 1324        STA A  P6        Store P6
02690 121B 7D 132A        TST    BHE       Bot.flag = 1 ?
02700 121E 26 04          BNE    GWP7
02710 1220 A6 81          LDA A  $81,X     No, set P7
02720 1222 20 03          BRA    GWP7+3
02730 1224 B6 131D GWP7   LDA A  EDGE      Yes, set edge
02740 1227 B7 1325        STA A  P7        Store P7
02750 122A 7D 1327        TST    LHE       L.flag = 1 ?
02760 122D 26 09          BNE    GWP8
02770 122F 7D 132A        TST    BHE       Bot.flag = 1 ?
02780 1232 26 04          BNE    GWP8
02790 1234 A6 80          LDA A  $80,X     No, set P8
02800 1236 20 03          BRA    GWP8+3
02810 1238 B6 131D GWP8   LDA A  EDGE      Yes, set edge
02820 123B B7 1326        STA A  P8        Store in P8
02830 123E 39             RTS
02840                *                 Main Variable Table :

02850 1300                ORG    $1300
02860 1300 0001   VDUNO   RMB    1
02870 1301 0001   BUFNO   RMB    1
02880 1302 B000   VDU0ST  FDB    VDUST
02890 1304 B020   VDU1ST  FDB    VDUST+$20
02900 1306 B400   VDU2ST  FDB    VDUST+$400
02910 1308 B420   VDU3ST  FDB    VDUST+$420
02920 130A 3E00   BUF0ST  FDB    BUFST
02930 130C 3F00   BUF1ST  FDB    BUFST+$100
02940 130E 0002   VDULOC  RMB    2
02950 1310 0002   BUFLOC  RMB    2
02960 1312 0002   DSUM1   RMB    2
02970 1314 0002   DSUM2   RMB    2
02980 1316 0002   DSUM3   RMB    2
02990 1318 0001   XCO     RMB    1
03000 1319 0001   YCO     RMB    1
03010 131A 0001   ZCO     RMB    1
03020 131B 0001   B2VDIR  RMB    1
03030 131C 0001   B2DDIR  RMB    1
03040 131D 0001   EDGE    RMB    1
03050 131E 0001   P0      RMB    1
03060 131F 0001   P1      RMB    1
03070 1320 0001   P2      RMB    1
03080 1321 0001   P3      RMB    1
03090 1322 0001   P4      RMB    1
03100 1323 0001   P5      RMB    1
03110 1324 0001   P6      RMB    1
03120 1325 0001   P7      RMB    1
03130 1326 0001   P8      RMB    1
03140 1327 0001   LHE     RMB    1
03150 1328 0001   THE     RMB    1
03160 1329 0001   RHE     RMB    1
03170 132A 0001   BHE     RMB    1
03180 132B 0003   ADDR    RMB    3
03190 132E 60     BIT0    FCB    $60
03200 132F FF     BIT1    FCB    $FF
03210 1330 0001   TNTK    RMB    1
03220 1331 0001   ORP     RMB    1
03230 1332 2E     PIT0    FCB    $2E
```

Assembly Listing of LPP Software Simulator

Part 3 - Sub-routine Facilities Section (Used by Parts 1 and 2)

```
03240 1333 4F        PIT1   FCB    $4F
03250 1334 3F        PITX   FCB    $3F
03260 1335 0001      PPL    RMB    1
03270 1336 0001      PPR    RMB    1
03280 1337 10        PRTON  FCB    $10
03290 1338 11        PRTOFF FCB    $11
03300 1339 01        NCRLF  FCB    $01
03310 133A 0002      SOURCE RMB    2
03320 133C 0002      DEST   RMB    2
03330 133E 0001      SRCNO  RMB    1
03340 133F 0001      INCR   RMB    1
03350 1340 0002      STS    RMB    2
03360 1342 0002      DTS    RMB    2
03370 1344 0001      VDUNOI RMB    1
03380 1345 0001      BUFNOI RMB    1
03390 1346 0002      SRCI   RMB    2
03400 1348 0002      STSI   RMB    2
03410 134A 0001      SRCNOI RMB    1
03420 134B 0001      INCRI  RMB    1
03430 134C 0001      VDUNOE RMB    1
03440 134D 0001      BUFNOE RMB    1
03450 134E 0002      SRCE   RMB    2
03460 1350 0002      STSE   RMB    2
03470 1352 0001      SRCNOE RMB    1
03480 1353 0001      INCRE  RMB    1
03490 1354 0001      VDUNOO RMB    1
03500 1355 0001      BUFNOO RMB    1
03510 1356 0002      DESTO  RMB    2
03520 1358 0002      DTSO   RMB    2
03530 135A 0002      SAVEX  RMB    2
03540 135C 0001      INCR1  RMB    1
03550 135D 0E        PDOUB  FCB    $0E
03560 135E 0F        PSING  FCB    $0F
03570 135F 3F        BITX   FCB    $3F
03580                *
03590 1400                  ORG    $1400
03600 1400 7F 131B V2B2D    CLR    B2VDIR   - VDU CHAR -
03610 1403 73 131B          COM    B2VDIR   - VDUX ) BUF ) DISK
03620 1406 7F 131C          CLR    B2DDIR     Set directions
03630 1409 BD 105A          JSR    B2V        Do V to B
03640 140C 7E 1041          JMP    B2D        Do B to D
03650                *
03660 140F BD A980 T2B      JSR    IPTIL    - TAPE CHAR ) BUF
03670 1412 BD A880          JSR    IPTAPE     Read tape,
03680 1415 BD A880          JSR    IPTAPE      ignoring leader
03690 1418 BD A880          JSR    IPTAPE      and 1st 4 chars.
03700 141B BD A880          JSR    IPTAPE
03710 141E CE 3E00          LDX    £BUFST     Get buffer addr.
03720 1421 5F               CLR B             Clear byte count
03730 1422 BD A880 T2B1     JSR    IPTAPE     Get char.
03740 1425 A7 00            STA A  0,X        Store in buffer
03750 1427 A7 80            STA A  $80,X      (upper and lower)
03760 1429 08               INX               Incr.buffer addr.
03770 142A 5C               INC B             Incr.byte count
03780 142B C1 80            CMP B  £$80       Count = 128 ?
03790 142D 26 F3            BNE    T2B1       No,loop back...
03800 142F 39               RTS               Yes,return.....
03810                *
03820 1430 7F 1335 PPAT     CLR    PPL      - PRINT TWO PATTERNS
03830 1433 7F 1336          CLR    PPR        Set LH,RH
03840 1436 7C 1336          INC    PPR         pattern nos.
03850 1439 B6 1337          LDA A  PRTON      Turn printer ON
03860 143C BD E1D1          JSR    OUTEEE
03870 143F 7F 1318          CLR    XCO        Clear x
03880 1442 B6 1335 PPAT2    LDA A  PPL        Get LH pat. no.
03890 1445 B7 1300          STA A  VDUNO
03900 1448 BD 14F9          JSR    PLINE      Print LH pat.line
03910 144B B6 1336          LDA A  PPR        Get RH pat. no.
03920 144E 81 01            CMP A  £$01
03930 1450 2E 06            BGT    PPAT1      Number ) 1 ?
03940 1452 B7 1300          STA A  VDUNO      No,
03950 1455 BD 1479          JSR    PLINE       print RH pat.line
03960 1458 BD E0B1 PPAT1    JSR    WAIT2      Do CRLF
03970 145B 7C 1318          INC    XCO        Increment x
03980 145E B6 1318          LDA A  XCO
03990 1461 81 20            CMP A  £$20       x = 32 ?
04000 1463 2D DD            BLT    PPAT2       No,loop back...
04010 1465 BD 146E          JSR    CRLFN       Yes,do Nx CRLFs
04020 1468 B6 1338          LDA A  PRTOFF
04030 146B 7E E1D1          JMP    OUTEEE      Turn printer OFF
04040                *
04050 146E 5F       CRLFN   CLR B             - 'CRLF' N TIMES
04060 146F BD E0B1          JSR    WAIT2       Do CRLF
04070 1472 5C               INC B             Incr. count
04080 1473 F1 1339          CMP B  NCRLF      Count = 'NCRLF' ?
04090 1476 26 F7            BNE    CRLFN+1    No, loop back....
04100 1478 39               RTS               Yes, return.....
04110                *
04120 1479 BD E0CC PLINE    JSR    OUTS     - PRINT LINE OF
04130 147C BD E0CC          JSR    OUTS     - PATTERN
04140 147F BD E0CC          JSR    OUTS
04150 1482 BD E0CC          JSR    OUTS       Print 5 spaces
04160 1485 BD E0CC          JSR    OUTS
04170 1488 7F 1319          CLR    YCO        Clear y
04180 148B BD 112E          JSR    XYLOC      Get VDU pointer
04190 148E A6 00            LDA A  0,X        Get VDU pixel
04200 1490 B1 132E          CMP A  BIT0       Is pixel 'BIT0' ?
04210 1493 27 0F            BEQ    PLINE1
04220 1495 B1 132F          CMP A  BIT1       Is pixel 'BIT1' ?
04230 1498 27 05            BEQ    *+7
04240 149A B6 1334          LDA A  PITX       No, get 'PITX'
04250 149D 20 08            BRA    PLINE2
04260 149F B6 1333          LDA A  PIT1       Get 'PIT1'
04270 14A2 20 03            BRA    PLINE2
04280 14A4 B6 1332 PLINE1   LDA A  PIT0       Get 'PIT0'
04290 14A7 BD E1D1 PLINE2   JSR    OUTEEE     Print char.
04300 14AA 7C 1319          INC    YCO        Increment y
04310 14AD B6 1319          LDA A  YCO
```

Assembly Listing of LPP Software Simulator

Part 3 - Sub-routine Facilities Section (Used by Parts 1 and 2)

```
04320 14B0 81 20          CMP A  £$20       Is y = 32 ?
04330 14B2 26 D7          BNE    PLINE+18   No, loop back...
04340 14B4 39             RTS
04350              *
04360 14B5 7F 131C D2B2V  CLR    B2DDIR - TRANSFER CHARACTER
04370 14B8 73 131C        COM    B2DDIR - DISK > BUF > VDU
04380 14BB BD 1041        JSR    B2D       Set directions,
04390 14BE 7F 131B        CLR    B2VDIR    Do D to B, then
04400 14C1 7E 105A        JMP    B2V        B to V.
04410              *
04420 14C4 BD 140F T2B2V  JSR    T2B     - TRANSFER CHARACTER
04430 14C7 20 F5          BRA    D2B2V+9 - TAPE > BUF > VDU
04440              *
04450 14C9 BD E0B1 TVREAD JSR    WAIT2     - READ CAMERA
04460 14CC CE 15D1        LDX    £XTVR1    - PICTURE TO VDU
04470 14CF BD E07E        JSR    PDATA1    "Satisfactory ? "
04480 14D2 86 02          LDA A  £$02      Then read in
04490 14D4 B7 D026        STA A  TVCW       picture.......
04500 14D7 B6 D027 TVR1   LDA A  TVCR
04510 14DA 48             ASL A
04520 14DB 2B FA          BMI    TVR1
04530 14DD F6 D020        LDA B  TVXMIN    Get x direction
04540 14E0 F7 15C7        STA B  XMIN       start/finish
04550 14E3 B6 D021        LDA A  TVXMAX
04560 14E6 10             SBA
04570 14E7 5F             CLR B
04580 14E8 44             LSR A            16 bit
04590 14E9 56             ROR B             right shift
04600 14EA 47             ASR A             five places(/32)
04610 14EB 56             ROR B             to :
04620 14EC 47             ASR A            Generate
04630 14ED 56             ROR B             'DX'
04640 14EE 47             ASR A             (x increment)
04650 14EF 56             ROR B
04660 14F0 47             ASR A
04670 14F1 56             ROR B
04680 14F2 B7 15C9        STA A  DX
04690 14F5 F7 15CA        STA B  DX+1
04700 14F8 F6 D022        LDA B  TVYMIN    Get y direction
04710 14FB F7 15C8        STA B  YMIN       start/finish
04720 14FE B6 D023        LDA A  TVYMAX
04730 1501 10             SBA
04740 1502 5F             CLR B
04750 1503 44             LSR A
04760 1504 56             ROR B
04770 1505 47             ASR A            16 bit
04780 1506 56             ROR B             right shift
04790 1507 47             ASR A             five places(/32)
04800 1508 56             ROR B             to :
04810 1509 47             ASR A            Generate
04820 150A 56             ROR B             'DY'
04830 150B 47             ASR A             (y increment)
04840 150C 56             ROR B
04850 150D B7 15CB        STA A  DY
04860 1510 F7 15CC        STA B  DY+1
04870 1513 B6 15C7        LDA A  XMIN
04880 1516 B7 15CD        STA A  XP
04890 1519 7F 15CE        CLR    XP+1
04900 151C 7F 1318        CLR    XCO       Clear x
04910 151F B6 15C8 TVR2   LDA A  YMIN
04920 1522 B7 15CF        STA A  YP
04930 1525 7F 15D0        CLR    YP+1
04940 1528 7F 1319        CLR    YCO       Clear y
04950 152B B6 15CD TVR3   LDA A  XP
04960 152E B7 D024        STA A  TVXP
04970 1531 B6 15CF        LDA A  YP
04980 1534 B7 D025        STA A  TVYP
04990 1537 86 01          LDA A  £$01
05000 1539 B7 D026        STA A  TVCW
05010 153C B6 1319        LDA A  YCO
05020 153F 5F             CLR B
05030 1540 47             ASR A
05040 1541 56             ROR B
05050 1542 47             ASR A
05060 1543 56             ROR B
05070 1544 36             PSH A
05080 1545 17             TBA
05090 1546 F6 1318        LDA B  XCO
05100 1549 01             NOP
05110 154A 1B             ABA
05120 154B 16             TAB
05130 154C 32             PUL A
05140 154D BB 1302        ADD A  VDUOST    Calculate VDU
05150 1550 B7 130E        STA A  VDULOC     address
05160 1553 F7 130F        STA B  VDULOC+1
05170 1556 FE 130E        LDX    VDULOC
05180 1559 B6 D027 TVR4   LDA A  TVCR       Sample ready ?
05190 155C 2B FB          BMI    TVR4       No, loop back....
05200 155E 36             PSH A             Yes, stack it...
05210 155F F6 1300        LDA B  VDUNO
05220 1562 27 07          BEQ    *+9        L or R VDU ?
05230 1564 FE 130E        LDX    VDULOC
05240 1567 BD 10AD        JSR    INX32      Adjust address
05250 156A FF 130E        STX    VDULOC      if necessary
05260 156D 32             PUL A
05270 156E 84 01          AND A  £$01       Get sample back
05280 1570 27 05          BEQ    TVR5
05290 1572 B6 132F        LDA A  BIT1       Convert to
05300 1575 20 03          BRA    TVR6        'BIT0 or 1'
05310 1577 B6 132E TVR5   LDA A  BIT0
05320 157A A7 00   TVR6   STA A  0,X        Store on VDU
05330 157C B6 15CF        LDA A  YP
05340 157F F6 15D0        LDA B  YP+1       Increment y
05350 1582 FB 15CC        ADD B  DY+1        by 'DY'
05360 1585 B9 15CB        ADC A  DY
05370 1588 B7 15CF        STA A  YP
05380 158B F7 15D0        STA B  YP+1
05390 158E 7C 1319        INC    YCO
```

Assembly Listing of LPP Software Simulator

Part 3 - Sub-routine Facilities Section
(Used by Parts 1 and 2)

```
05400 1591 B6 1319          LDA A  YCO
05410 1594 81 20            CMP A  £$20       y finished ?
05420 1596 27 03            BEQ    TVR7
05430 1598 7E 152B          JMP    TVR3
05440 159B B6 15CD TVR7     LDA A  XP         Yes...
05450 159E F6 15CE          LDA B  XP+1       Increment x
05460 15A1 FB 15CA          ADD B  DX+1        by 'DX'
05470 15A4 B9 15C9          ADC A  DX
05480 15A7 B7 15CD          STA A  XP
05490 15AA F7 15CE          STA B  XP+1
05500 15AD 7C 1318          INC    XCO
05510 15B0 B6 1318          LDA A  XCO
05520 15B3 81 20            CMP A  £$20       x finished ?
05530 15B5 27 03            BEQ    TVR8
05540 15B7 7E 151F          JMP    TVR2       Yes.....
05550 15BA BD E1AC TVR8     JSR    INEEE      Get response to
05560 15BD 81 59            CMP A  £'Y        "O.K. ? "
05570 15BF 27 03            BEQ    TVR9       Is it Y ?
05580 15C1 7E 14D2          JMP    TVREAD+9   No,do again....
05590 15C4 7E E0B1 TVR9     JMP    WAIT2      Yes,return....
05600                *
05610       D020   TVXMIN EQU    $D020      Machine addresses
05620       D021   TVXMAX EQU    $D021      of Low Resolution
05630       D022   TVYMIN EQU    $D022      interface
05640       D023   TVYMAX EQU    $D023      registers
05650       D024   TVXP   EQU    $D024
05660       D025   TVYP   EQU    $D025
05670       D026   TVCW   EQU    $D026
05680       D027   TVCR   EQU    $D027
05690                *
05700 15C7 0001    XMIN   RMB    1          Variables used
05710 15C8 0001    YMIN   RMB    1           in above routine
05720 15C9 0002    DX     RMB    2
05730 15CB 0002    DY     RMB    2
05740 15CD 0002    XP     RMB    2
05750 15CF 0002    YP     RMB    2
                                             Text strings :

05760 15D1 53      XTVR1  FCC    'SATISFACTORY CAMERA IMAGE
                                  ("Y" OR X) ? '
05770 15F8 04             FCB    4
05780 15F9 53      XSTAS  FCC    'STARTING TRACK AND SECTOR
                                  (TTOS) ? '
05790 161C 04             FCB    4
05800 161D 4E      XSRCNO FCC    'NO. OF SOURCE CHARACTERS
                                  (HH) ? '
05810 163D 04             FCB    4
05820 163E 49      XINCR  FCC    'INCREMENTAL STEP IN SOURCE
                                  FILE (HH) ? '
05830 1665 04             FCB    4
05840 1666 20      XSINIT FCC    ' - SOURCE OF DATA - '
05850 167A 0D             FCB    $0D,$0A
05860 167C 54             FCC    'TAPE , DISK , CAMERA ,
                                  X(DUMMY) ? '
05870 169E 04             FCB    4
05880 169F 20      XDINIT FCC    ' - DESTINATION OF DATA - '
05890 16B8 0D             FCB    $0D,$0A
05900 16BA 44             FCC    'DISK , X(DUMMY) ? '
05910 16CC 04             FCB    4
05920                *
05930 1700                ORG    $1700
05940 1700 FF 135A PREP   STX    SAVEX      - PRINT CRLF,
05950 1703 BD E0B1        JSR    WAIT2      - PROMPT STRING,
05960 1706 FE 135A        LDX    SAVEX      - ACCEPT CHAR IN B
05970 1709 BD E07E        JSR    PDATA1     - PRINT CRLF.
05980 170C BD E1AC        JSR    INEEE
05990 170F 16             TAB
06000 1710 7E E0B1        JMP    WAIT2
06010                *
06020 1713 CE 15F9 GETTS  LDX    £XSTAS     - GET STARTING
06030 1716 BD E07E        JSR    PDATA1     - TRACK AND SECTOR
06040 1719 7E E047        JMP    BADDR      - INTO X
06050                *
06060 171C 08      INXTS  INX               - 'INX' IN FORMAT
06070 171D FF 135A        STX    SAVEX      - FOR TRACK AND
06080 1720 B6 135B        LDA A  SAVEX+1    - SECTOR
06090 1723 84 F0          AND A  £$F0
06100 1725 27 09          BEQ    INXTS1     Carry into bit4 ?
06110 1727 7F 135B        CLR    SAVEX+1    Yes,incr.next
06120 172A 7C 135A        INC    SAVEX       byte
06130 172D FE 135A        LDX    SAVEX
06140 1730 39      INXTS1 RTS
06150                *
06160 1731 FF 135A SINIT1 STX    SAVEX      - INIT SOURCE AND
06170 1734 BD E0B1        JSR    WAIT2      - RELEVANT
06180 1737 FE 135A        LDX    SAVEX      - PARAMETERS
06190 173A BD E07E        JSR    PDATA1      "(Section)"
06200 173D CE 1666        LDX    £XSINIT
06210 1740 BD 1709        JSR    PREP+9      "Source of data?"
06220 1743 C1 54          CMP B  £'T
06230 1745 26 07          BNE    S1          Is it T ?
06240 1747 CE 14C4        LDX    £T2B2V
06250 174A FF 133A S3     STX    SOURCE      Yes,'tape'...
06260 174D 39             RTS
06270 174E C1 44   S1     CMP B  £'D         Is it D ?
06280 1750 26 0E          BNE    S4
06290 1752 BD 1713        JSR    GETTS       Yes,get T+S
06300 1755 FF 1340        STX    STS          from terminal
06310 1758 BD E0B1        JSR    WAIT2
06320 175B CE 1485        LDX    £D2B2V      And store'disk'..
06330 175E 20 EA          BRA    S3
06340 1760 C1 43   S4     CMP B  £'C         Is it C ?
06350 1762 26 05          BNE    S7
06360 1764 CE 14C9        LDX    £TVREAD     Yes,'camera'...
06370 1767 20 E1          BRA    S3
06380 1769 C1 58   S7     CMP B  £'X         Is it X ?
06390 176B 26 D0          BNE    SINIT1+12   Illegal - ignore
06400 176D CE 100F        LDX    £DUMMY      Yes,'dummy'....
```

Assembly Listing of LPP Software Simulator

Part 3 - Sub-routine Facilities Section
(Used by Parts 1 and 2)

```
06410 1770 20 D8          BRA    S3
06420              *
06430 1772 CE 161D SINIT2 LDX    £XSRCNO   - GET NO.OF SOURCE
06440 1775 BD E07E        JSR    PDATA1    - CHARS.
06450 1778 BD E055        JSR    BYTE        "No.?",I/P byte
06460 177B B7 133E        STA A  SRCNO         from terminal
06470 177E 39             RTS                  to 'SRCNO'
06480              *
06490 177F B6 133E SINIT3 LDA A  SRCNO     - GET INCREMENTAL
06500 1782 4A             DEC A            - IF NECESSARY
06510 1783 2F 0F          BLE    S11         Is No.= 1 ?
06520 1785 BD E081        JSR    WAIT2       No,
06530 1788 CE 163E        LDX    £XINCR       "Increment ?"
06540 178B BD E07E        JSR    PDATA1
06550 178E BD E055        JSR    BYTE        I/P byte
06560 1791 B7 133F        STA A  INCR         to 'INCR'
06570 1794 7E E0B1 S11    JMP    WAIT2
06580              *
06590 1797 FF 135A DINIT  STX    SAVEX     - INIT DEST AND
06600 179A BD E081        JSR    WAIT2     - AND RELEVANT
06610 179D FE 135A        LDX    SAVEX     - PARAMTERS -
06620 17A0 BD E07E        JSR    PDATA1      "(Section)"
06630 17A3 CE 169F        LDX    £XDINIT     "Destination ?"
06640 17A6 BD 1709        JSR    PREP+9
06650 17A9 20 05          BRA    D2
06660 17AB FF 133C D5     STX    DEST
06670 17AE 20 20          BRA    D10
06680 17B0 C1 44   D2     CMP B  £'D         Is it D ?
06690 17B2 26 11          BNE    D3
06700 17B4 CE 1400        LDX    £V2B2D      Yes,'disk'...
06710 17B7 FF 133C        STX    DEST
06720 17BA BD 1713        JSR    GETTS       Get T+S
06730 17BD FF 1342        STX    DTS
06740 17C0 BD E0B1        JSR    WAIT2
06750 17C3 20 0B          BRA    D10
06760 17C5 C1 58   D3     CMP B  £'X         Is it X ?
06770 17C7 26 05          BNE    D4          Illegal - ignore
06780 17C9 CE 100F        LDX    £DUMMY      Yes,'dummy'....
06790 17CC 20 DD          BRA    D5
06800 17CE 20 C7   D4     BRA    DINIT
06810 17D0 39      D10    RTS
06820              *
06830 1800                ORG    $1800
06840 1800 FE 1344 GIPPP  LDX    VDUNOI - GET IPP PARAMETERS
06850 1803 FF 1300        STX    VDUNO
06860 1806 FE 1346        LDX    SRCI     - from IPP stores,
06870 1809 FF 133A        STX    SOURCE
06880 180C FE 1348        LDX    STSI       to general
06890 180F FF 1340        STX    STS          stores
06900 1812 FE 134A        LDX    SRCNOI
06910 1815 FF 133E GIPPP1 STX    SRCNO
06920 1818 39             RTS
06930              *
06940 1819 FE 134C GEXPP  LDX    VDUNOE - GET EXP PARAMETERS
06950 181C FF 1300        STX    VDUNO
06960 181F FE 134E        LDX    SRCE     - from EXP stores,
06970 1822 FF 133A        STX    SOURCE
06980 1825 FE 1350        LDX    STSE       to general
06990 1828 FF 1340        STX    STS          stores
07000 182B FE 1352        LDX    SRCNOE
07010 182E 20 E5          BRA    GIPPP1
07020              *
07030 1830 FE 1354 GOPPP  LDX    VDUNOO - GET OPP PARAMETERS
07040 1833 FF 1300        STX    VDUNO
07050 1836 FE 1356        LDX    DESTO    - from OPP stores,
07060 1839 FF 133C        STX    DEST
07070 183C FE 1358        LDX    DTSO       to general
07080 183F FF 1342        STX    DTS          stores
07090 1842 39             RTS
07100              *
07110 1843 FE 1300 SIPPP  LDX    VDUNO    - STORE IPP
07120 1846 FF 1344        STX    VDUNOI   - PARAMETERS
07130 1849 FE 133A        LDX    SOURCE
07140 184C FF 1346        STX    SRCI      - from general
07150 184F FE 1340        LDX    STS             stores,
07160 1852 FF 1348        STX    STSI
07170 1855 FE 133E        LDX    SRCNO       to IPP stores
07180 1858 FF 134A        STX    SRCNOI
07190 185B 39             RTS
07200              *
07210 185C FE 1300 SEXPP  LDX    VDUNO    - STORE EXP
07220 185F FF 134C        STX    VDUNOE   - PARAMETERS
07230 1862 FE 133A        LDX    SOURCE
07240 1865 FF 134E        STX    SRCE      - from general
07250 1868 FE 1340        LDX    STS             stores,
07260 186B FF 1350        STX    STSE
07270 186E FE 133E        LDX    SRCNO       to EXP stores
07280 1871 FF 1352        STX    SRCNOE
07290 1874 39             RTS
07300              *
07310 1875 FE 1300 SOPPP  LDX    VDUNO    - STORE OPP
07320 1878 FF 1354        STX    VDUNOO   - PARAMETERS
07330 187B FE 133C        LDX    DEST
07340 187E FF 1356        STX    DESTO     - from general
07350 1881 FE 1342        LDX    DTS             stores,
07360 1884 FF 1358        STX    DTSO
07370 1887 39             RTS                  to OPP stores
07380              *
07400                     END

TOTAL ERRORS 00000

ENTER PASS : 1P,2P,2L,2T
```

APPENDIX 2a Sheet 1 of 4 M.M. Listings

Memory Matrix Cell Addresses and Corresponding Features

for F5 Machine (3x3 bit Window Format)

512 Cells with Addresses in Range 000-1FF (Hex)

'Preferred Subset' of Cells are boxed

Pairs of Cells with Crossing Number of 2 are asterisked

000	001	002 *	003	004 *	005	006 *	007	008	009	00A *	00B	00C	00D	00E *	00F
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
010	011	012	013	014	015	016	017	018	019	01A *	01B	01C *	01D	01E *	01F
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
020	021	022	023	024	025	026	027	028	029	02A *	02B	02C	02D	02E *	02F
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
030	031	032	033	034	035	036	037	038	039	03A	03B	03C	03D	03E *	03F
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
040	041	042	043	044	045	046	047	048	049	04A	04B	04C	04D	04E	04F
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X
050	051	052	053	054	055	056	057	058	059	05A	05B	05C	05D	05E	05F
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X
060	061	062	063	064	065	066	067	068	069	06A *	06B	06C *	06D	06E *	06F
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X
070	071	072	073	074	075	076	077	078	079	07A *	07B	07C *	07D	07E *	07F
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X	..X

Memory Matrix Cell Addresses and Corresponding Features

for F5 Machine (3x3 bit Window Format)

512 Cells with Addresses in Range 000-1FF (Hex)

'Preferred Subset' of Cells are boxed

Pairs of Cells with Crossing Number of 2 are asterisked

080	081	082	083	084	085	086	087	088	089	08A	08B	08C	08D	08E	08F
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.

090	091	092	093	094	095	096	097	098	099	09A	09B	09C	09D	09E	09F
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.

0A0	0A1	0A2	0A3	0A4	0A5	0A6	0A7	0A8	0A9	0AA	0AB	0AC	0AD	0AE	0AF
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.

0B0	0B1	0B2	0B3	0B4	0B5	0B6	0B7	0B8	0B9	0BA	0BB	0BC	0BD	0BE	0BF
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.	.X.

0C0	0C1	0C2	0C3	0C4	0C5	0C6	0C7	0C8	0C9	0CA	0CB	0CC	0CD	0CE	0CF
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX

0D0	0D1	0D2	0D3	0D4	0D5	0D6	0D7	0D8	0D9	0DA	0DB	0DC	0DD	0DE	0DF
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX

0E0	0E1	0E2	0E3	0E4	0E5	0E6	0E7	0E8	0E9	0EA	0EB	0EC	0ED	0EE	0EF
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX

0F0	0F1	0F2	0F3	0F4	0F5	0F6	0F7	0F8	0F9	0FA	0FB	0FC	0FD	0FE	0FF
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX	.XX

Memory Matrix Cell Addresses and Corresponding Features

for F5 Machine (3x3 bit Window Format)

512 Cells with Addresses in Range 000-1FF (Hex)

'Preferred Subset' of Cells are boxed

Pairs of Cells with Crossing Number of 2 are asterisked

100	101	102	103	104	105	106	107	108	109	10A	10B	10C	10D	10E	10F
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..

110	111	112	113	114	115	116	117	118	119	11A	11B	11C	11D	11E	11F
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..

120	121	122	123	124	125	126	127	128	129	12A	12B	12C	12D	12E	12F
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..

130	131	132	133	134	135	136	137	138	139	13A	13B	13C	13D	13E	13F
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..	X..

140	141	142	143	144	145	146	147	148	149	14A	14B	14C	14D	14E	14F
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X

150	151	152	153	154	155	156	157	158	159	15A	15B	15C	15D	15E	15F
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X

160	161	162	163	164	165	166	167	168	169	16A	16B	16C	16D	16E	16F
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X

170	171	172	173	174	175	176	177	178	179	17A	17B	17C	17D	17E *	17F
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X	X.X

Memory Matrix Cell Addresses and Corresponding Features

for F5 Machine (3x3 bit Window Format)

512 Cells with Addresses in Range 000-1FF (Hex)

'Preferred Subset' of Cells are boxed

Pairs of Cells with Crossing Number of 2 are asterisked

180	181	182	183	184	185	186	187	188	189	18A	18B	18C	18D	18E	18F
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.

190	191	192	193	194	195	196	197	198	199	19A	19B	19C	19D	19E	19F
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.

1A0	1A1	1A2	1A3	1A4	1A5	1A6	1A7	1A8	1A9	1AA	1AB	1AC	1AD	1AE	1AF
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.

1B0	1B1	1B2	1B3	1B4	1B5	1B6	1B7	1B8	1B9	1BA	1BB	1BC	1BD	1BE	1BF
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.	XX.

1C0	1C1	1C2	1C3	1C4	1C5	1C6	1C7	1C8	1C9	1CA	1CB	1CC	1CD	1CE	1CF
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX

1D0	1D1	1D2	1D3	1D4	1D5	1D6	1D7	1D8	1D9	1DA	1DB	1DC	1DD	1DE	1DF
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.	...	.X.	X..	XX.
XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX

1E0	1E1	1E2	1E3	1E4	1E5	1E6	1E7	1E8	1E9	1EA	1EB	1EC	1ED	1EE	1EF
...	...	...	...	X..	X..	X..	X..	.X.	.X.	.X.	.X.	XX.	XX.	XX.	XX.
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX

1F0	1F1	1F2	1F3	1F4	1F5	1F6	1F7	1F8	1F9	1FA	1FB	1FC	1FD	1FE	1FF
..X	..X	..X	..X	X.X	X.X	X.X	X.X	.XX	.XX	.XX	.XX	XXX	XXX	XXX	XXX
..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX	..X	.XX	X.X	XXX
XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX

APPENDIX 2b Sheet 1 of 2 M.M. Listings

Memory Matrix Cell Contents from F5 Machine

Trained to 'THIN' using Rot.+Ref. in Experiment 8

512 Cells with Addresses in Range 000-1FF (Hex)

'Preferred Subset' of Cells are boxed

000 .	001 ?	002 ?	003 ?	004 .	005 ?	006 .	007 ?	008 ?	009 ?	00A ?	00B ?	00C .	00D ?	00E .	00F ?
010 .	011 ?	012 ?	013 ?	014 ?	015 ?	016 .	017 ?	018 .	019 ?	01A ?	01B ?	01C .	01D ?	01E .	01F .
020 ?	021 ?	022 ?	023 ?	024 ?	025 ?	026 ?	027 ?	028 ?	029 ?	02A ?	02B ?	02C ?	02D ?	02E ?	02F ?
030 .	031 ?	032 ?	033 ?	034 .	035 ?	036 ?	037 ?	038 .	039 ?	03A ?	03B ?	03C .	03D .	03E ?	03F .
040 .	041 ?	042 ?	043 ?	044 ?	045 ?	046 ?	047 ?	048 ?	049 ?	04A ?	04B ?	04C ?	04D ?	04E ?	04F ?
050 ?	051 ?	052 ?	053 ?	054 ?	055 ?	056 ?	057 ?	058 .	059 ?	05A ?	05B ?	05C ?	05D ?	05E ?	05F ?
060 .	061 ?	062 ?	063 ?	064 ?	065 ?	066 ?	067 ?	068 ?	069 ?	06A ?	06B ?	06C ?	06D ?	06E ?	06F ?
070 .	071 ?	072 ?	073 ?	074 ?	075 ?	076 ?	077 ?	078 .	079 .	07A ?	07B ?	07C .	07D .	07E .	07F .

080 ?	081 ?	082 ?	083 ?	084 ?	085 ?	086 ?	087 ?	088 ?	089 ?	08A ?	08B ?	08C ?	08D ?	08E ?	08F ?
090 ?	091 ?	092 ?	093 ?	094 ?	095 ?	096 ?	097 ?	098 ?	099 ?	09A ?	09B ?	09C ?	09D ?	09E ?	09F ?
0A0 ?	0A1 ?	0A2 ?	0A3 ?	0A4 ?	0A5 ?	0A6 ?	0A7 ?	0A8 ?	0A9 ?	0AA ?	0AB ?	0AC ?	0AD ?	0AE ?	0AF ?
0B0 ?	0B1 ?	0B2 ?	0B3 ?	0B4 ?	0B5 ?	0B6 ?	0B7 ?	0B8 ?	0B9 ?	0BA ?	0BB ?	0BC ?	0BD ?	0BE ?	0BF ?
0C0 .	0C1 ?	0C2 ?	0C3 ?	0C4 ?	0C5 ?	0C6 ?	0C7 ?	0C8 ?	0C9 ?	0CA ?	0CB ?	0CC ?	0CD ?	0CE ?	0CF ?
0D0 .	0D1 ?	0D2 ?	0D3 ?	0D4 ?	0D5 ?	0D6 ?	0D7 ?	0D8 ?	0D9 ?	0DA ?	0DB ?	0DC ?	0DD ?	0DE ?	0DF ?
0E0 .	0E1 ?	0E2 ?	0E3 ?	0E4 ?	0E5 ?	0E6 ?	0E7 ?	0E8 ?	0E9 ?	0EA ?	0EB ?	0EC ?	0ED ?	0EE ?	0EF ?
0F0 .	0F1 .	0F2 ?	0F3 ?	0F4 ?	0F5 ?	0F6 ?	0F7 ?	0F8 ?	0F9 .	0FA ?	0FB ?	0FC .	0FD .	0FE ?	0FF .

APPENDIX 2b Sheet 2 of 2 M.M. Listings

Memory Matrix Cell Contents from F5 Machine

Trained to 'THIN' using Rot.+Ref. in Experiment 8

512 Cells with Addresses in Range 000-1FF (Hex)

'Preferred Subset' of Cells are boxed

100	101	102	103	104	105	106	107	108	109	10A	10B	10C	10D	10E	10F
.	?	.	?	?	?	.	?	?	?	?	?	.	?	.	.
110	111	112	113	114	115	116	117	118	119	11A	11B	11C	11D	11E	11F
?	?	?	?	?	?	?	?	?	?	?	?	?	?	.	.
120	121	122	123	124	125	126	127	128	129	12A	12B	12C	12D	12E	12F
?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
130	131	132	133	134	135	136	137	138	139	13A	13B	13C	13D	13E	13F
?	?	?	?	?	?	?	?	?	?	?	?	?	?	.	.
140	141	142	143	144	145	146	147	148	149	14A	14B	14C	14D	14E	14F
?	?	.	?	?	?	?	?	?	?	?	?	?	?	?	?
150	151	152	153	154	155	156	157	158	159	15A	15B	15C	15D	15E	15F
?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
160	161	162	163	164	165	166	167	168	169	16A	16B	16C	16D	16E	16F
.	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
170	171	172	173	174	175	176	177	178	179	17A	17B	17C	17D	17E	17F
?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	.

180	181	182	183	184	185	186	187	188	189	18A	18B	18C	18D	18E	18F
.	?	.	?	.	?	.	.	?	?	?	?	?	?	?	.
190	191	192	193	194	195	196	197	198	199	19A	19B	19C	19D	19E	19F
?	?	?	?	?	?	?	?	?	?	?	?	?	?	.	.
1A0	1A1	1A2	1A3	1A4	1A5	1A6	1A7	1A8	1A9	1AA	1AB	1AC	1AD	1AE	1AF
?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
1B0	1B1	1B2	1B3	1B4	1B5	1B6	1B7	1B8	1B9	1BA	1BB	1BC	1BD	1BE	1BF
?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	.
1C0	1C1	1C2	1C3	1C4	1C5	1C6	1C7	1C8	1C9	1CA	1CB	1CC	1CD	1CE	1CF
.	?	.	.	?	?	.	.	?	?	?	?	?	?	.	.
1D0	1D1	1D2	1D3	1D4	1D5	1D6	1D7	1D8	1D9	1DA	1DB	1DC	1DD	1DE	1DF
?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	.
1E0	1E1	1E2	1E3	1E4	1E5	1E6	1E7	1E8	1E9	1EA	1EB	1EC	1ED	1EE	1EF
.	.	?	.	?	?	.	.	?	?	?	?	?	?	?	.
1F0	1F1	1F2	1F3	1F4	1F5	1F6	1F7	1F8	1F9	1FA	1FB	1FC	1FD	1FE	1FF
.	.	.	.	?	?	?	.	.	.	?	.	?	.	?	X

APPENDIX 2c Sheet 1 of 2 M.M. Listings

Memory Matrix Cell Contents from F5 Machine

Trained to 'CLEAN UP' using Rot.+Ref. in Experiment 11

100 Pairs of Characters used in Training Set

512 Cells with Addresses in Range 000-1FF (Hex)

Cells set to '?' (Initialisation State) are boxed

000	001	002	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F
.	.	.	.	.	.	.	X	.	.	.	X	.	X	.	.
010	011	012	013	014	015	016	017	018	019	01A	01B	01C	01D	01E	01F
.	.	.	.	.	.	.	.	.	X	.	X	.	.	.	X
020	021	022	023	024	025	026	027	028	029	02A	02B	02C	02D	02E	02F
.	.	.	.	.	.	.	X	.	X	?	.	.	X	.	.
030	031	032	033	034	035	036	037	038	039	03A	03B	03C	03D	03E	03F
.	X	.	X	.	.	.	X	.	.	.	.	.	X	.	.
040	041	042	043	044	045	046	047	048	049	04A	04B	04C	04D	04E	04F
.	.	.	.	.	.	.	?	.	.	?	?	.	?	?	?
050	051	052	053	054	055	056	057	058	059	05A	05B	05C	05D	05E	05F
.	.	?	?	.	.	.	?	.	.	?	?	.	X	?	X
060	061	062	063	064	065	066	067	068	069	06A	06B	06C	06D	06E	06F
.	X	.	X	.	?	?	?	.	X	?	X	?	X	?	X
070	071	072	073	074	075	076	077	078	079	07A	07B	07C	07D	07E	07F
.	.	.	?	.	X	X	X	.	X	?	X	.	X	.	X

080	081	082	083	084	085	086	087	088	089	08A	08B	08C	08D	08E	08F
.	.	.	X	.	.	.	X	.	.	?	.	.	X	.	.
090	091	092	093	094	095	096	097	098	099	09A	09B	09C	09D	09E	09F
.	.	?	?	?	?	?	?	.	X	?	X	.	?	?	X
0A0	0A1	0A2	0A3	0A4	0A5	0A6	0A7	0A8	0A9	0AA	0AB	0AC	0AD	0AE	0AF
.	X	?	.	?	?	?	X	?	.	?	?	?	X	.	X
0B0	0B1	0B2	0B3	0B4	0B5	0B6	0B7	0B8	0B9	0BA	0BB	0BC	0BD	0BE	0BF
.	X	?	X	?	?	?	?	.	.	.	X	?	X	.	X
0C0	0C1	0C2	0C3	0C4	0C5	0C6	0C7	0C8	0C9	0CA	0CB	0CC	0CD	0CE	0CF
.	X	.	X	.	?	?	X	.	X	?	X	?	?	?	X
0D0	0D1	0D2	0D3	0D4	0D5	0D6	0D7	0D8	0D9	0DA	0DB	0DC	0DD	0DE	0DF
.	.	?	?	.	?	?	?	.	X	?	?	X	X	?	X
0E0	0E1	0E2	0E3	0E4	0E5	0E6	0E7	0E8	0E9	0EA	0EB	0EC	0ED	0EE	0EF
.	.	.	.	?	?	?	X	.	.	.	X	?	X	X	X
0F0	0F1	0F2	0F3	0F4	0F5	0F6	0F7	0F8	0F9	0FA	0FB	0FC	0FD	0FE	0FF
.	X	?	X	?	X	?	X	.	.	.	X	.	X	X	X

APPENDIX 2c Sheet 2 of 2 M.M. Listings

Memory Matrix Cell Contents from F5 Machine

Trained to 'CLEAN UP' using Rot.+Ref. in Experiment 11

100 Pairs of Characters used in Training Set

512 Cells with Addresses in Range 000-1FF (Hex)

Cells set to '?' (Initialisation State) are boxed

100	101	102	103	104	105	106	107	108	109	10A	10B	10C	10D	10E	10F
.	.	.	X	.	.	.	.	.	.	.	X	.	.	.	X
110	111	112	113	114	115	116	117	118	119	11A	11B	11C	11D	11E	11F
.	.	.	?	.	.	.	X	.	?	?	X	.	X	.	X
120	121	122	123	124	125	126	127	128	129	12A	12B	12C	12D	12E	12F
.	.	.	X	?	?	.	?	?	?	?	X	?	?	?	X
130	131	132	133	134	135	136	137	138	139	13A	13B	13C	13D	13E	13F
.	?	?	?	.	?	X	X	?	?	?	X	?	X	.	X
140	141	142	143	144	145	146	147	148	149	14A	14B	14C	14D	14E	14F
.	.	.	.	.	.	.	X	?	?	?	?	.	?	?	X
150	151	152	153	154	155	156	157	158	159	15A	15B	15C	15D	15E	15F
.	.	.	?	?	?	?	?	.	?	?	?	?	?	?	X
160	161	162	163	164	165	166	167	168	169	16A	16B	16C	16D	16E	16F
.	.	.	X	.	?	X	X	?	?	?	?	?	?	?	X
170	171	172	173	174	175	176	177	178	179	17A	17B	17C	17D	17E	17F
.	X	X	X	?	?	?	X	?	X	?	X	?	X	.	X

180	181	182	183	184	185	186	187	188	189	18A	18B	18C	18D	18E	18F
.	X	.	.	.	.	.	X	.	X	.	.	.	X	.	.
190	191	192	193	194	195	196	197	198	199	19A	19B	19C	19D	19E	19F
.	?	?	?	.	?	?	X	?	?	?	X	X	X	.	X
1A0	1A1	1A2	1A3	1A4	1A5	1A6	1A7	1A8	1A9	1AA	1AB	1AC	1AD	1AE	1AF
.	X	.	.	?	?	?	X	?	X	.	X	?	?	.	X
1B0	1B1	1B2	1B3	1B4	1B5	1B6	1B7	1B8	1B9	1BA	1BB	1BC	1BD	1BE	1BF
?	X	?	X	?	?	?	X	?	X	X	X	?	X	X	X
1C0	1C1	1C2	1C3	1C4	1C5	1C6	1C7	1C8	1C9	1CA	1CB	1CC	1CD	1CE	1CF
.	.	.	X	.	X	.	X	.	?	?	X	X	X	.	X
1D0	1D1	1D2	1D3	1D4	1D5	1D6	1D7	1D8	1D9	1DA	1DB	1DC	1DD	1DE	1DF
.	X	?	X	?	?	?	X	X	X	?	X	?	X	.	X
1E0	1E1	1E2	1E3	1E4	1E5	1E6	1E7	1E8	1E9	1EA	1EB	1EC	1ED	1EE	1EF
.	X	.	.	?	X	.	X	?	X	.	X	?	X	X	X
1F0	1F1	1F2	1F3	1F4	1F5	1F6	1F7	1F8	1F9	1FA	1FB	1FC	1FD	1FE	1FF
.	X	.	X	?	X	.	X	.	X	X	X	.	X	X	X

APPENDIX 2d Sheet 1 of 2 M.M. Listings

Memory Matrix Cell Contents from F5 Machine

Trained to 'EDGE FIND' using Rot.+Ref. in Experiment 12

This M.M. was 'keyed in' by operator and is consequently

NOT the result of training by example

512 Cells with Addresses in Range 000-1FF (Hex)

'Preferred Subset' of Cells are boxed

000	001	002	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F
.	X	.	X	.	X	.	X	.	X	.	X	.	X	.	X
010	011	012	013	014	015	016	017	018	019	01A	01B	01C	01D	01E	01F
.	X	.	X	.	X	.	X	.	X	.	X	.	X	.	X
020	021	022	023	024	025	026	027	028	029	02A	02B	02C	02D	02E	02F
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	X
030	031	032	033	034	035	036	037	038	039	03A	03B	03C	03D	03E	03F
.	X	.	X	.	X	.	X	.	X	.	X	.	X	.	X
040	041	042	043	044	045	046	047	048	049	04A	04B	04C	04D	04E	04F
.	X	.	X	.	X	.	X	.	X	.	X	.	X	.	X
050	051	052	053	054	055	056	057	058	059	05A	05B	05C	05D	05E	05F
.	X	.	X	.	X	.	X	.	X	.	X	.	X	.	X
060	061	062	063	064	065	066	067	068	069	06A	06B	06C	06D	06E	06F
.	X	.	X	.	X	.	X	.	X	.	.	.	.	.	.
070	071	072	073	074	075	076	077	078	079	07A	07B	07C	07D	07E	07F
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	X

080	081	082	083	084	085	086	087	088	089	08A	08B	08C	08D	08E	08F
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	X
090	091	092	093	094	095	096	097	098	099	09A	09B	09C	09D	09E	09F
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	.
0A0	0A1	0A2	0A3	0A4	0A5	0A6	0A7	0A8	0A9	0AA	0AB	0AC	0AD	0AE	0AF
.	X	.	.	.	X	.	.	.	.	.	.	.	.	.	.
0B0	0B1	0B2	0B3	0B4	0B5	0B6	0B7	0B8	0B9	0BA	0BB	0BC	0BD	0BE	0BF
.	X	.	.	.	X	.	.	.	X	.	.	.	.	.	.
0C0	0C1	0C2	0C3	0C4	0C5	0C6	0C7	0C8	0C9	0CA	0CB	0CC	0CD	0CE	0CF
.	X	.	X	.	X	.	.	.	X	.	.	.	X	.	.
0D0	0D1	0D2	0D3	0D4	0D5	0D6	0D7	0D8	0D9	0DA	0DB	0DC	0DD	0DE	0DF
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	X
0E0	0E1	0E2	0E3	0E4	0E5	0E6	0E7	0E8	0E9	0EA	0EB	0EC	0ED	0EE	0EF
.	X	.	X	.	X	.	.	.	X	.	.	.	.	.	.
0F0	0F1	0F2	0F3	0F4	0F5	0F6	0F7	0F8	0F9	0FA	0FB	0FC	0FD	0FE	0FF
.	X	.	.	.	X	.	X	.	X	.	.	.	X	.	.

Memory Matrix Cell Contents from F5 Machine

Trained to 'EDGE FIND' using Rot.+Ref. in Experiment 12

This M.M. was 'keyed in' by operator and is consequently

NOT the result of training by example

512 Cells with Addresses in Range 000-1FF (Hex)

'Preferred Subset' of Cells are boxed

100	101	102	103	104	105	106	107	108	109	10A	10B	10C	10D	10E	10F
.	X	.	X	.	X	.	X	.	X	.	X	.	X	.	X
110	111	112	113	114	115	116	117	118	119	11A	11B	11C	11D	11E	11F
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	X
120	121	122	123	124	125	126	127	128	129	12A	12B	12C	12D	12E	12F
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	.
130	131	132	133	134	135	136	137	138	139	13A	13B	13C	13D	13E	13F
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	X
140	141	142	143	144	145	146	147	148	149	14A	14B	14C	14D	14E	14F
.	X	.	X	.	X	.	X	.	X	.	X	.	X	.	X
150	151	152	153	154	155	156	157	158	159	15A	15B	15C	15D	15E	15F
.	X	.	X	.	X	.	X	.	X	.	X	.	X	.	X
160	161	162	163	164	165	166	167	168	169	16A	16B	16C	16D	16E	16F
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	X
170	171	172	173	174	175	176	177	178	179	17A	17B	17C	17D	17E	17F
.	X	.	X	.	X	.	X	.	X	.	X	.	X	.	X

180	181	182	183	184	185	186	187	188	189	18A	18B	18C	18D	18E	18F
.	X	.	X	.	X	.	X	.	X	.	X	.	X	.	X
190	191	192	193	194	195	196	197	198	199	19A	19B	19C	19D	19E	19F
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	X
1A0	1A1	1A2	1A3	1A4	1A5	1A6	1A7	1A8	1A9	1AA	1AB	1AC	1AD	1AE	1AF
.	X	.	X	.	X	.	.	.	.	.	.	.	.	.	.
1B0	1B1	1B2	1B3	1B4	1B5	1B6	1B7	1B8	1B9	1BA	1BB	1BC	1BD	1BE	1BF
.	.	.	.	.	X	.	X	.	.	.	.	.	X	.	.
1C0	1C1	1C2	1C3	1C4	1C5	1C6	1C7	1C8	1C9	1CA	1CB	1CC	1CD	1CE	1CF
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	X
1D0	1D1	1D2	1D3	1D4	1D5	1D6	1D7	1D8	1D9	1DA	1DB	1DC	1DD	1DE	1DF
.	X	.	X	.	X	.	X	.	X	.	X	.	X	.	X
1E0	1E1	1E2	1E3	1E4	1E5	1E6	1E7	1E8	1E9	1EA	1EB	1EC	1ED	1EE	1EF
.	X	.	X	.	X	.	X	.	.	.	.	.	X	.	.
1F0	1F1	1F2	1F3	1F4	1F5	1F6	1F7	1F8	1F9	1FA	1FB	1FC	1FD	1FE	1FF
.	X	.	X	.	X	.	X	.	X	.	.	.	X	.	.