SOME ASPECTS OF ADAPTIVE LOGIC FOR PATTERN RECOGNITION

by

C. Y. E. CHEUNG-YUENG-SAN

Thesis presented for the Degree of Doctor of Philosophy

in the Faculty of Natural Sciences

University of Kent

CANTERBURY

1973

ABSTRACT

This thesis deals with pattern recognizers (PRs) which are adaptive and amenable to hardware realization. Such PRs consist of networks of microcircuit modules (SLAMs: Stored-Logic Adaptive Microcircuits) which are used as feature extractors and which ensure a high throughput by their parallel operation.

Previous workers have adopted the number of training patterns as a measure for training such PRs on different pattern classes. In this thesis, the number of memory bits set in the SLAMs is considered instead, and this is shown to provide a better balance between sections of the PRs called discriminators.

Simulations are carried out to observe how the size of the PR and the amount of training affect the performance and a quantitative comparison with a template-matching classifier is presented. The effect of clustering the SLAM inputs within the input matrix is also investigated and PRs with SLAMs which have their outputs weighted according to their memory contents are also simulated.

From the results, a technique to optimize the performance of the PRs is proposed and a possible development of the SLAM module is suggested.

CONTENTS

CHAPTER 1

INTRODUCTION & BACKGROUND

1.1 Introduction

The object of this thesis is to investigate the possibilities of building a pattern recognition machine, using 'Stored-Logic Adaptive Microcircuit' (SLAM) modules (Aleksander & Albrow, 1968a) as feature extractors. A feature in this work is defined as a subset of the possible states of a binary n-tuple (sampled n points of a binary pattern) to which a SLAM is connected (Ullmann, 1968). The SLAMs are trained on such features of known patterns and when presented with a subsequent unknown pattern the SLAMs indicate the presence or absence of the previously seen features. The pattern recognition machine then classifies the unknown pattern on the basis of the SLAMs' responses. The experimentation in this thesis has been restricted to handwritten numerals but the method should be applicable to pattern recognition in general and to related disciplines where pattern recognition concepts are used.

Although much has been written on the problem of pattern recognition, no unifying concept or general theory is as yet evident. This is partly because the problem, being so broad, has not been clearly defined and mostly

because it is not yet understood how human beings perform the task, a task which is done with seeming ease in their daily lives. As digital computers and electronic machines grow in numbers and increase in speed, the necessity of building machines to take over from human operators becomes more apparent, especially in areas where the human operators may be lacking in speed, accuracy or low cost.

Reading of handwritten characters, recognition of a spoken word independent of the speaker who utters it, recognition of a speaker regardless of the spoken text, interpretation of electrocardiograms, recognition of cloud patterns for weather forecasting, etc.... are only a few of the problems that have so far remained unsolved. It can be argued that almost any field of scientific or human activity deals in terms of pattern recognition. The scientific field of pattern recognition in general concerns itself with the solution of these problems. All of these problems, however different as they may seem, have a common requirement for their solutions. The common requirement is to possess the ability of recognizing membership of classes and hence a criterion for differentiating between members of different classes.

In the absence of an all-embracing theory, time and again during the design of a pattern recognition machine, it is necessary for the designer to make educated guesses.

This often makes the design of pattern recognizers less of an exact science. The chief sources of specification of many pattern recognition models being intuition and introspection.

Many pattern recognition systems are completely determined in advance in the sense of a fixed complete design. There may have been adaptation by virtue of an evolution of algorithms based on the designer's experience with the system. They are however basically non-adaptive, as opposed to adaptive systems which adjust parameters or modify algorithms.

For the purpose of this thesis a class of patterns is defined as a set of binary vectors which in some useful way can be treated alike. A number of assumptions have been made in the body of the report. To make it clear when a restrictive assumption is introduced, the word 'assume' will be used and underlined.

Having briefly outlined the motivation for pattern recognition machines, the organization of the thesis is now given.

In the remaining part of chapter 1, a pattern recognition system is conceptually described and the factors that influence the design of a pattern recognizer

(PR) are given.

Various pattern recognition techniques are briefly surveyed in chapter 2, and they are broadly classified as follows:- correlation, statistical, discriminant function and feature-extraction methods.

In chapter 3 a PR using SLAMs is described and a criterion for training such PR is defined.

Chapter 4 is entirely devoted to experimentation carried out in this work. PRs using SLAMs are simulated and the performance is compared with that of a 'Template-Matching' classifier. PRs using SLAMs with weighted outputs are also simulated.

In chapter 5 the salient points of the experimentation are revisited and possible developments are suggested.

Finally, in chapter 6 a quantitative comparison is made, followed by a summary of the whole work and some concluding remarks.

1.2 Phases of Operation of an Adaptive Pattern Recognizer

The operation of an adaptive PR may be divided into two phases: the Learning or Training phase and the Recognition or Testing phase. The author is aware of the controversies that arise in conjunction with the words 'learning' and 'training'. However, they are used here in a technical sense and are not meant to imply any parallelism with living organisms.

In the recognition phase, the machine makes decisions as to which of some previously specified classes of patterns the unknown patterns belong. One of the pattern classes could be a 'reject' class which by definition contains patterns not belonging to any of the specified classes. The recognition phase is the useful phase where the machine actually performs work by classifying patterns. Before the PR can perform reliably, it must have been trained to detect somehow the significant features of the pattern classes; hence a set of patterns which is assumed to be representative of the unknown patterns is used in the training phase.

1.3 The Layers of a Pattern Recognizer

A PR can generally be divided into 3 layers as shown in fig. 1.1. It consists of a Sensory Layer or Retina on

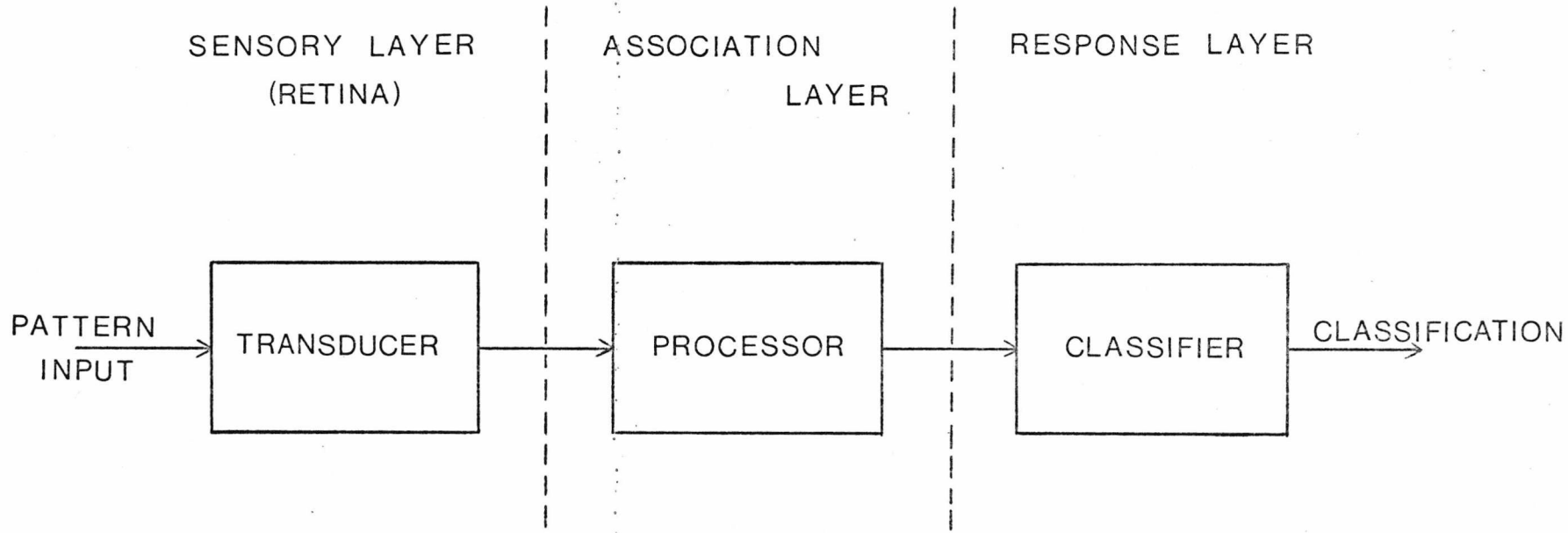

FIGURE 1.1

Block Diagram of a Pattern Recognizer

which the pattern to be recognized is implanted, an Association Layer or Processor where the pattern is re-mapped or described in terms of its characteristic features, and a Response Layer or Classifier where the verdict is finally given as to which class of patterns the unknown pattern belongs.

The retina is the transducer which receives and digitizes the analogue pattern, it can be a photodiode matrix or a Vidicon camera or a CRT flying spot camera. The digitization is equivalent to placing a matrix on the pattern and putting a binary bit '1' in the squares where the black area exceeds a certain threshold and a binary bit '0' in the remaining squares. The operation therefore reduces the analogue pattern X into a set of d real variables $x_1,\ldots,x_d$ called the Pattern Vector and the individual variables are the components. As the variables in this case can only be either '1' or '0', the pattern can be represented by a vertex called a Pattern Point in a d-dimensional hypercube, which is referred to as Pattern Space or Pattern Hypercube.

The processor stage is where the feature extraction process takes place. It may be highly elaborate, involving several layers of investigation in itself, or it may be simply an assessment of the activity of groups of retinal points, called n-tuples. The process is almost invariably

accompanied with reduction in dimensionality of the pattern vector, yielding a set of numbers $f_1,\ldots,f_s$ that constitute the input to the classifier. The s numbers are the components of the Feature Vector, which now describes the pattern and can again be talked of geometrically as a Feature Point in an s-dimensional Feature Space or Feature Hypercube if the variables can have only binary values of '1' or '0'.

The processor stage effectively transforms the pattern space into a feature space for the classifier to operate upon. The classifier, perhaps according to a preprogrammed rule, partitions the s-dimensional space into disjoint regions, each region associated with only one pattern class. The rule is equivalent to a decision function representing a decision surface, or a set of decision surfaces. The decision surface may be implicitly defined as in the case of 'Pandemonium' (Selfridge, 1959). With the latter machine the *a posteriori* probabilities of class membership are computed and the pattern is classified as belonging to the most probable class.

The decision surface may also be explicitly defined as a set of geometric hyperplanes in the feature space, dividing the space into compartments. The unlabelled pattern is then classified as belonging to the class corresponding to the compartment in which its feature point

is located. In this report the rule is assumed to be non-randomized, meaning that the classifier always makes the same decision if the same feature point $F = (f_1,....,f_s)$ is being presented several times.

The greatest diversity in the pattern recognition machines designed to date lies in the design of the processor and the classifier. The main part of this report will be centered on the possible adaptive networks for a processor. Before the various processors and algorithms are reviewed, it is instructive to summarize the factors that influence the design of a pattern recognition machine.

1.4 Factors that Influence the Design of a Pattern Recognizer

The task of designing a PR is simplified by first making a few reasonable restrictive assumptions. In the following, the ith. pattern class is denoted by C_i, $i = 1,2,...,R$ for an R-class recognition problem. With each class of patterns is associated a number of significant constants, the values of which may or may not be known.

(i) The *a priori* probability, P(i), that an unknown

pattern is a member of C_i, rather than a member of one of the (R-1) remaining pattern classes. It is assumed in this work that the occurrence of the patterns is according to some time-stationary first order probability, and that they are not intentionally arranged in any manner. This assumption is not always satisfied as in some cases the patterns shows a Markov-dependence (Raviv, 1967), or as in the case of letters in a text there are strong statistical relationships between adjacent letters (Thomas & Kassler, 1967).

(ii) The cost, $K(j|i)$, entails in misclassifying an unlabelled pattern which is really a member of C_i as being a member of C_j. In the following it is assumed that $K(j|i)$ does not change with the percentage of misclassified members of C_i.

(iii) The cost, $K(i)$, arises when a pattern belonging to class C_i is rejected, that is when the recognition system witholds its recognition decision and the pattern is rejected for exceptional handling, such as a rescan or manual inspection. The option to reject is introduced to safeguard against excessive misclassification; it converts potential misrecognition into rejection. On the other hand, some would-be correct classifications are also converted into rejections. It is assumed here that

the cost K(i) is independent of the number of patterns rejected.

(iv) Gradual changes with time in the class distributions can happen due to data changes or hardware degradation. If this takes place, then the parameter values in the pattern recognition machine must be updated to follow the changes (Amari, 1967). It may usually be assumed as in the following that the distributions of the patterns are stationary in time.

(v) The labelled patterns of each class are truly representative members of the class. With certain kinds of patterns e.g. medical records such as ECGs, labelling or classification of the patterns cannot be done with 100% accuracy. Sometimes only a representative collection of unlabelled patterns is available (Chien & Fu, 1967). However in this work, as much care as possible has been taken in the collection and the processing of the data so that member patterns remain representative, and it is assumed that the patterns which are taken to be representative of members of C_i are indeed representative of C_i members.

(vi) The nature of relationship between the members of the same class is another important factor. The members of each class can be a typical pattern, a prototype, corrupted by noise as in a typewritten font; or

the members of each class are a collection of well-defined, noisefree patterns, which have certain features in common, as perfectly noisefree letters may be drawn in a great many ways (Akers & Rutter, 1964); or even the class members may have certain features in common while at the same time they are somewhat corrupted by noise, which is usually the case in handwritten script. The latter case is assumed in this thesis.

(vii) The class C_i can be made up of several distinct subclasses, C_{i1}, C_{i2}, etc...; the distribution of the C_i pattern vectors will then be multimodal in the pattern space, i.e. with local maxima or even disjointed subspaces. Such a multimodal multivariate distribution can create difficult problems in some classification procedures like partitioning of the pattern space by hyperplanes as will be discussed in section 2.2.3.

Here one deals only with handwritten numerals, not with patterns which have been synthesized by computer. This restriction also excludes pictorial representation of three spatial dimensions (stereopairs, contour maps, etc...) as well as time-varying pictorial information (e.g. on line character recognition in real time); thus concerning only with two-dimensional numerals.

These restrictions have been imposed for the following reasons:

(i) Convenience. The data is available and the familiarity with handwritten numbers is exploited.

(ii) Background. Research on alphanumeric pattern recognition has been vigorously pursued, and use of the relatively large amount of literature on the subject can be made.

(iii) Usefulness. Any success can be immediately put into use, and the techniques can be applied to other pattern recognition problems.

1.5 Assessment of a Pattern Recognizer's Performance

It is necessary to estimate how well a PR will recognize patterns before it is put to use, or before it is built if the machine is being simulated as in this case. The usual procedure to obtain such an estimate is to train (or design) the recognizer using representative patterns: the design or training patterns. Its performance is evaluated on a separate set of representative patterns: the test patterns. The result is then considered as an estimate of the PR capability.

CHAPTER 2

PRELIMINARY PROCESSING AND VARIOUS PATTERN RECOGNITION TECHNIQUES

2.1 Preliminary Processing

It is assumed that the main operation in the retina (fig. 1.1) is to transform the analogue pattern into discrete form and quite often it is preceded as well as followed by some form of preprocessing. The first step being object isolation, as other patterns and the background tends to obscure the pattern in question. The segmentation of speech is as yet an unsolved problem. The recognition of chromosomes and blood cells is greatly hampered by shortcomings in this area. However, in simpler tasks such as reading typescript or handwritten text, *a priori* knowledge of the recognition problem can be used, e.g.

(i) Suitable filtering may be used to enhance the signal to noise ratio for waveforms.

(ii) Handwritten alphanumeric characters may be normalized with respect to size, orientation and texture.

(iii) The pattern can be centralised in the digitizing frame.

The quantizing process itself results in a loss of information concerning the original pattern; if the grid

is too coarse, certain details of the pattern will be lost, but this loss can be reduced by having more squares in the grid. However, there is a limit to the amount of useful grid resolution and the quantizing process can sometimes be useful in filtering out insignificant irregularities that might cause confusion. For example, if a character printed by a typewriter is magnified a few thousand times, it is generally somewhat difficult to identify it because of the apparently random distribution of tiny blobs of ink on the paper.

Loss of information is also incurred by having only 2 intensity levels. These simplifications are partly traded against less complexity in subsequent stages of the PR. Other preliminary processing that can be carried out are:

(iv) Thinning of the character strokes.

(v) Filling in isolated holes in otherwise black areas.

(vi) Filling in small notches in straight edge segments.

(vii) Eliminating isolated '1' bits.

(viii) Eliminating small bumps along straight edge segments.

(ix) Replacing missing corner points.

(i) to (iv) are particularly useful in recognition of handwritten characters where the size and thickness of strokes differ considerably (Genchi *et al.*, 1968).

(v) to (ix) are smoothing processes to remove random noise in the patterns and to smooth any irregularities that may be introduced by the quantization.

Almost all the preprocessing techniques use the 'window' method: a small submatrix or window is centered in turn on each retinal point and the latter is transformed onto a new matrix depending on the contents of the submatrix. A comparative assessment of various preprocessing schemes has been made in Deutsch (1969), where all the transformations are prespecified and intuitively designed. Adaptively generated transformations have also been used (Saraga & Woollons, 1968).

2.2 Various Techniques Used in Pattern Recognition

In the following, the various techniques used in pattern recognition are described. It is by no means a comprehensive review but only a description of the techniques that have influenced the present work in some way or other.

2.2.1 Correlation Methods

The techniques of correlation in two dimensions can

be used to compare a standard or representative pattern with an unlabelled pattern. Given R representatives, one for each class, the unlabelled pattern is correlated with each of them and it is classified in the same class as the same representative for which the correlation is a maximum. This type of approach is often termed 'Template Matching'.

Although the concept seems straightforward, these methods using either digital or optical means, present certain difficulties in their application (Highleyman, 1961; Selfridge & Neisser, 1963). Sensitivity to translation, magnification and orientation are characteristic of the approach. In order to achieve a match with a stored reference item, it is necessary to prenormalize the pattern and eliminate these effects. This is often difficult since different examples of the same class may not really have similar shapes. In fact, even if they do, slight variations may result in poor correlation with the stored template. Highleyman (1961) has suggested two methods for reducing position sensitivity in the case of character recognition:

(i) To align the centre of gravity of the unknown pattern with that of the reference template.

(ii) To generate a two dimensional correlation matrix by shifting with respect to the

template until a maximum correlation is achieved.

The first method is faster and does not allow for the possibility of convergence to an incorrect maximum.

If it is known that only one template will match the unlabelled pattern, and if the search is done sequentially, the search can be stopped when such a template has been found. On the average R/2 templates must be checked before the pattern is identified. The correlation can, however, be done in parallel by optical means.

If there is more than one prototype per class, the same correlation techniques are used with all the prototypes and the unlabelled pattern is classified as the prototype which gives the maximum correlation. This method is sometimes called 'Nearest Neighbour Classification' (Cover, 1967).

The template matching is also referred to as Minimum Distance Classification as the prototypes can be represented by pattern points in a pattern space and any unknown pattern is implicitly classified according to the nearest prototype in the pattern space. It is also well-known that minimizing a Euclidean distance is equivalent to minimizing a Hamming distance.

All the above correlation techniques can be implemented by linear decision functions (Highleyman, 1962 ; Nilsson, 1965). In general, although the techniques are conceptually attractive, they are optimal only under certain very restrictive symmetry conditions on the distributions of the patterns. Consequently they have not been very useful except in cases of typed and printed characters which are very stylized and mainly corrupted with random noise.

2.2.2 Statistical Methods

In situations where the patterns in the various classes (R) are governed by multivariate probability density functions and that the latter can be estimated over the whole d-dimensional pattern, then Bayes' Criterion can be used as a classification rule. It is assumed that there is a sufficiently large number of representative patterns and that it is possible to obtain an accurate estimation of all the probability density functions which are:

(i) The *a priori* probabilities P(i), i = 1,...,R. P(i) is the probability of occurrence of a pattern from class C_i.

(ii) The conditional probabilities P(X|i),

$i = 1,\ldots,R$. $P(X|i)$ is the probability of occurrence of pattern X, given that it belongs to C_i.

If the cost or risk associated with each classification is known,

$$K(i|j) \quad \text{for } i, j = 1,\ldots,R.$$

where $K(i|j)$ is the cost of classifying a pattern to class i, when it really belongs to class j. If $P(j|X)$ is the probability that, given X, its category is C_j, then the conditional average cost for deciding that a particular pattern belongs to class i is given by

$$L(i) = \sum_{j=1}^{R} K(i|j)P(j|X) \tag{2.1}$$

The average cost is minimized if the classifier always assigned X to class i when

$$L(i) \leq L(j) \quad \text{for } j = 1,\ldots,R.$$

By Bayes' rule

$$P(j|X) = \frac{P(X|j)P(j)}{P(X)} \tag{2.2}$$

Substitution of Eq. (2.2) into Eq. (2.1) yields

$$L(i) = \frac{1}{P(X)} \sum_{j=1}^{R} K(i|j)P(X|j)P(j)$$

In the computation of $L(i)$ for $i = 1,\ldots,R$, the

quantity $1/P(X)$ occurs as a common factor; therefore the value of i that minimizes L(i) for any given X also minimizes

$$L'(i) = \sum_{j=1}^{R} K(i|j)P(X|j)P(j) \qquad (2.3)$$

The above decision rule which minimizes the expected loss is called Bayes' Criterion or Bayes' strategy. Other statistical methods are:

(i) The Maximum Likelihood method which is used in cases where the classification costs and the *a priori* probabilities are unknown.

(ii) The Minimax method which is applicable when the *a priori* probabilities are unknown.

(iii) The Neyman-Pearson method which is applicable where the *a priori* probabilities and the classification costs are not known and furthermore $R = 2$.

A detailed discussion of the above statistical methods is given in Andrews (1972).

In practice, however, the statistical approach

seldom works out. There are usually too few representative patterns available to determine the multivariate distributions, or there is not enough time and funding to process the many patterns even if they are available. If the patterns are of size 10x10 and of binary components, it becomes necessary to estimate R probabilities at 2^{100} points. A certain form for the probability function is usually assumed; a common assumption is that given a certain pattern class the measurements made by the receptor are normally distributed and that each measurement is independent of the others.

Another practical difficulty is that unless the separation surfaces are of fairly simple shape or the dimensionality of the pattern space is low, the number of constants needed to describe the surfaces becomes so large that information storage in the classifier presents problems.

2.2.3 Discriminant Function Methods

Separation of a pattern space can be achieved by separation surfaces like hyperplanes. The location of such surfaces in a d-dimensional space is determined by the values of (d+1) constants or coefficients; for example, the equation for a hyperplane in a d-dimensional space is given by

$$g(X) = 0$$

where $g(X) = w_1x_1 + w_2x_2 + \ldots + w_dx_d + w_{d+1}$

$w_1, w_2, \ldots, w_{d+1}$ are the (d+1) coefficients

x_i is the coordinate in the ith. dimension.

and g(X) is called a Discriminant Function.

For a two-class classification, the discriminant function gives a positive value to any point on the positive side of the hyperplane and a negative value to any point on the negative side. Such a function can be implemented by a Threshold Logic Unit as shown in fig. 2.1, consisting of weights, a summing device and a threshold element.

The idea of a threshold logic unit with adjustable weights is the embodiment of two concepts; the first concept is the on-off threshold device which is a simplified model of a neuron proposed by McCulloch and Pitts (1943); the second is that long-term memory in animals depends on changes in the synaptic junctions between neurons (Hebb, 1949).

For an R-class pattern classifier, a set of R discriminant functions, $g_1(X), g_2(X), \ldots, g_R(X)$, is required. These functions are chosen such that for all pattern points in class C_i,

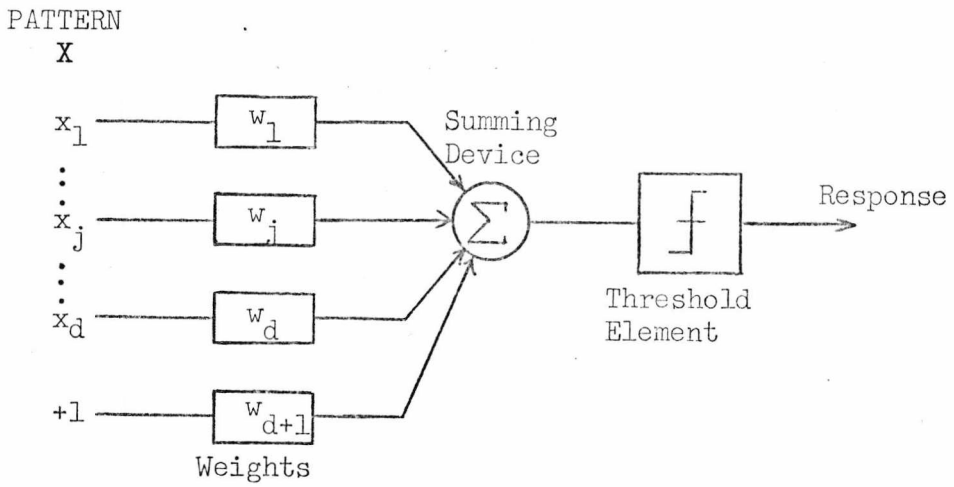

Figure 2.1. The Threshold Logic Unit.

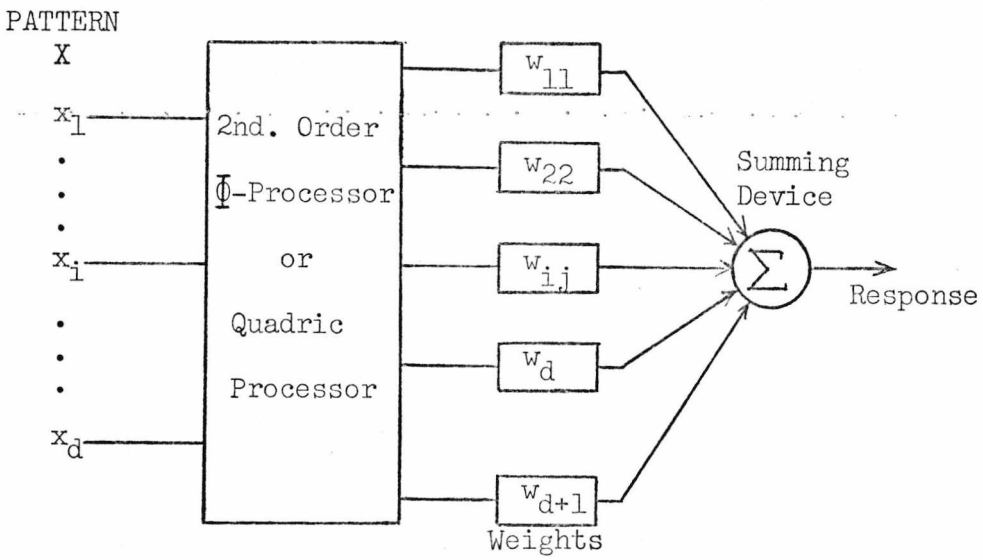

Figure 2.2. A Quadric Discriminator.

$$g_i(X) > g_j(X) \quad \text{for } j = 1,\ldots,R \text{ and } j \neq i.$$

Such a decision can be implemented by R discriminators, one for each discriminant function, followed by a maximum selector (Nilsson, 1965).

The limitation of the hyperplanes generated by threshold logic unit, is that they can perform only linear separations. In cases where the classes are non-linearly separable, higher order surfaces in the pattern space are more useful.

2.2.4 Φ-Discriminator

An nth. Φ-discriminator is capable of generating an nth. order polynomial surface in a d-dimensional space. It consists of a Φ-processor followed by weighting values and a summer. The input to the Φ-processor is the pattern vector X and the output is the vector Y whose components $f_i(X)$ are linearly independent, real, single-valued functions of X. The components $f_i(X)$ are given by,

(i) for a 1st. order Φ-processor which would generate linear decision surfaces,

$$f_i(X) = x_i \qquad i = 1,\ldots,d$$

(ii) for a 2nd. order Φ-processor which would generate quadric decision surfaces

$$f_i(X) = x_i^a \, x_j^b$$

$$i,j = 1,\ldots,d, \text{ and } a,b = 0 \ \& \ 1$$

(iii) for an nth. order Φ-processor

$$f_i(X) = \sum_{j=1}^{n} x_{p_j}^{a_j} \qquad (2.4)$$

$$p_j = 1,..,d \quad \text{and} \quad a_j = 0 \ \& \ 1$$

The discriminant function for an nth. order Φ-processor becomes

$$g(X) = \sum_{i=1}^{m} w_i f_i$$

An nth. order Φ-processor, in effect, converts the vector X into all the possible combinations (m) of its components up to nth. polynomial, as in fig. 2.2, which illustrates a quadric discriminator. A pattern classifying machine employing a Φ-discriminator is called a Φ-machine.

2.2.4.1 Parametric Training Methods

In some pattern recognition problems, the pattern classes are characterized by sets of parameters. Parametric training methods are those that use the training patterns to establish estimates of the values of the parameters and then use the estimates for the specification of the discriminant functions.

A situation in which the pattern classes are characterized by sets of parameters occurs when the patterns in each of the R classes are random variables governed by R distinct probability functions as in section 2.2.2. The optimum classification by Bayes' Criterion is given by Eq. 2.3.

$$L'(i) = \sum_{j=1}^{R} K(i|j)P(X|j)P(j) \qquad (2.3)$$

If the values of the parameters $P(X|j)$ and $P(j)$ for $j = 1,...R$, can be estimated from the training patterns, then such an optimum decision rule can be implemented using Φ-discriminators (Nilsson, 1965, chapter 3).

2.2.4.2 Non-parametric Training Methods

When no assumptions can be made about the distributions of the patterns, non-parametric training methods are used. Depending on the expected modalities of the distributions, some functional form for the separation is assumed, such as a linear, quadric or nth. order surface.

The values of the weights, before training begins, may be preset to any convenient values or they may be set to values selected at random. During the training phase, the training patterns are presented to the discriminator one at a time and the actual response is

compared with the desired response; if the classifier responds correctly to a pattern, no adjustments to the weights are made, otherwise the weights are adjusted. The various adjusting procedures are called error-correcting training procedures. The following two procedures would converge to a suitable hyperplane after a finite number of iterations, provided the pattern classes are linearly separable (Rosenblatt, 1960; Nilsson, 1965).

(i) Fixed Increment rule

The weights are altered by a fixed increment either by addition or subtraction and the adjustments may or may not result into a correct response.

(ii) Absolute Correction rule

The weights are adjusted just enough to achieve a correct response.

2.2.5 Feature Extraction Methods

In some pattern recognition problems, the patterns in the different classes are so intermixed that only a highly nonlinear method can separate them to the required degree of accuracy. In such cases the recognition process is sometimes divided into two stages, the first

consists of simplifying the problem sufficiently to render it tractable for the second.

The design of the first, or feature extraction stage, may be approached from two points of view. Either one attempts to transform the pattern space in such a manner that the members of each class exhibit less variability and the relative separation between the classes is increased, thus allowing the use of a simpler decision rule, or one reduces the dimensionality of the pattern space, permitting the application of more complicated decision schemes. Ideally one could accomplish both objectives with the same transformation, but unfortunately the transformation for the first objective generally increase the dimensionality.

The selection of an effective set of features is probably the most important step in the design of a PR using feature extraction methods. It is also the most difficult decision as there is no general theory to define what features are relevant for a particular problem. The amount of literature on this topic is overwhelming: Levine (1969), Minsky (1963), Nagy (1969), Uhr (1966), being typical examples. Thus, only a simple overview of the subject is given here.

The chief theoretical difficulty in feature extraction is that the features must be evaluated in terms of the decision stage rather than on their own. Convenient numerical criteria like percent misclassification can only be used to evaluate the whole system.

It is generally agreed that the features should be easier to store than the complete pattern and independent of commonly encountered forms of distortion like skew, size deformation, or other noise effects. Such invariant features can be generated by taking moments (Hu, 1962), or by integral geometry (Tenery, 1963) for features which are invariant with respect to the rotation and the translation of the patterns.

A concept about features which is intuitively appealing is the concept of connectivity, whereby features are localized and are defined as simple geometrical shapes, such as straight lines, edges, arcs, etc.... A lot of effort and ingenuity has been applied over the years to devising such intuitive features for example,

(i) Bomba (1959) uses features such as a diagonal line

(ii) Unger (1959) uses features such as horizontal and vertical cavities.

(iii) Genchi *et al.* (1968) use features such as horizontal and vertical strokes.

Simple geometric features which are usually prespecified can be detected with templates matching (mask-type threshold) devices. Topological information such as the number of line segments encountered by a slice of specified orientation through the pattern and the existence of bays, enclosures, indentations and symmetric conditions can be obtained by means of logical tests on a digital computer (Doyle, 1960).

In cases where the tests for the presence or absence of a feature are expensive, the tests may be performed one at a time and each new test is selected on the outcome of the previous tests. The sequential tests are halted when the estimated error probability reaches a preset threshold, or when they are exhausted. Such sequential-detection techniques have been described in the literature (Fu & Chien, 1967; Fu *et al.* 1967).

Since the features favoured by different investigators are seldom compared on the same data sets, objective evaluation of the merits of the different systems is difficult.

Another school of researchers base their features on n-tuples; an n-tuple is a set of n retinal points or

components of the pattern vector. According to Bledsoe and Browning (1959), the sub-pattern or the state of an n-tuple constitutes a feature, and an n-tuple shifted into a different position on the retina is regarded as a different n-tuple.

In Block's 'Perceptron' (1962) and Widrow's 'Adaline' (1960) the n retinal elements in an n-tuple are connected to a threshold logic unit. The alternative sub-patterns on that n-tuple which cause the threshold unit to give output '1', are regarded as constituting a feature.

In these instances the pattern space has been randomly transformed into a feature space with a significant decrease in dimensionality, and improvement has been obtained despite the fact that features are chosen randomly. An exhaustive search for the optimum n-tuples is unrealistic, for instance, restricting to sets of only 5 measurements on 4-tuples on a 10x10 matrix, there are

$$\left(\binom{100}{4}\right) \simeq 10^{30} \text{ distinct sets,}$$

which excludes the possibility of an exhaustive search algorithm. The optimal selection of the random features however remains an unsolved problem. Considering the characteristics of the SLAM module, which is described in chapter 3, the Bledsoe and Browning definition is

opted for. Whereas Bledsoe and Browning recommend exclusive, or non-overlapping, n-tuples, this is not necessarily so in the work described here.

2.3 Classification of Patterns from More than 2 Classes

Where the partitioning of the pattern space is implicit, the design for a PR for cases R (pattern classes) = 2 and R >2is the same in principle. The difference between the two cases is that a larger system may be needed as the number of pattern classes increases, e.g. in template matching, more templates to represent all the classes are required and for feature extraction methods, more feature extractors (discriminators) are used.

When the partitioning of the pattern space is explicit like the dichotomy by a hyperplane generated by a discriminant function, several techniques can be used to reduce the R-class classification into a two-class classification:

(i) The R classes are separated by R hyperplanes, with each hyperplane separating the members of one class from the members of the other (R-1) classes. This technique can be considered if it is known that members

of the same class form a tight cluster in the pattern space, and that the Euclidean distances between members of different classes are large compared to the largest 'cluster-radius'.

(ii) The R classes are separated pairwise (Highleyman, 1962). This technique requires R(R-1)/2 hyperplanes. A pattern is considered a member of class C_i when its pattern point lies on the C_i-side of each of the (R-1) hyperplanes that separate members of C_i from members of the remaining (R-1) classes.

(iii) The R-class classification can be reduced to a sequence of 2-class classification by first separating the R classes into 2 superclasses. The superclasses are then further sub-divided and the process repeated till individual classes are obtained (Unger, 1959). With this technique the number of hyperplanes (N_H) required is bounded by the following expression

$$R-1 \geq N_H \geq \text{Log}_2 R$$

With this method, some planning for the organization of the sequential decision tree is required.

2.4 Handwritten-Character Recognition

All the above techniques have been applied to

handwritten character recognition, on their own or in various combinations. Most of the literature on this problem relates to laboratory research and preprototype development, and virtually none of the recognizers for handwritten characters has been used in commercial applications. A striking exception is found in Japan (Harmon, 1972).

Most of the Japanese mail is hand addressed and intensive effort has been put into reading handwritten zip codes. Fully automatic mail sorters have been in service in Japan for some time (Genchi *et al.* 1968). The recognition rate of a single digit (written in preprinted boxes but otherwise unconstrained) for a large sample of letters averages 95%, which is a significant accomplishment. The recognition is achieved by extracting a sequence of simple geometrical features in horizontal zones of the character after normalization of the height of the character and the width of the strokes. No new technical innovations are responsible, just concentrated engineering development and economic pressure.

A typical exploratory study of recognition of handwritten characters uses linear decision function to categorize handwritten numerals (Highleyman, 1962). The hyperplanes which represent the decision function are determined from samples of the numerals and the

recognition rate of the system on a different set of samples is 61.6%.

Recognition of machine printed characters has been more successful for obvious reasons, e.g. Coombes (1972) has obtained 97% recognition rate on a test set of 1000 machine printed characters in multifont, representing 20 alphabetic-character classes. He proposes a systematic search of n-tuples and weighting values for threshold logic units which would linearly dichotomize the feature space.

CHAPTER 3

STORED-LOGIC ADAPTABLE MICROCIRCUIT (SLAM)

3.1 Description of a SLAM Module

A SLAM module (Aleksander & Albrow, 1968b) is an adaptive logic gate which can be taught to give a desirable binary output to a set of binary inputs. Fig. 3.1 illustrates the operation of a 3-input SLAM. An n-input SLAM module has n input terminals, a decoder and storage flip-flops or memory elements, any one of which is associated to one of the possible sets of inputs by the decoder. The desired response to a particular set of inputs is written into the respective memory element by a strobe at the Teach Clock input, while applying to the Teach input terminal the desired logical output.

In the absence of any teach input the device is a logical '0' or '1' depending upon the content of the particular storage element that is addressed by the binary inputs. An n-input SLAM is therefore capable of providing all the logical functions relating the inputs to the output i.e. 2^{2^n}, and can achieve all the possible dichotomies of the input configurations.

Set and Reset input terminals to the memory elements are provided to set them to give a '1' or '0' respectively,

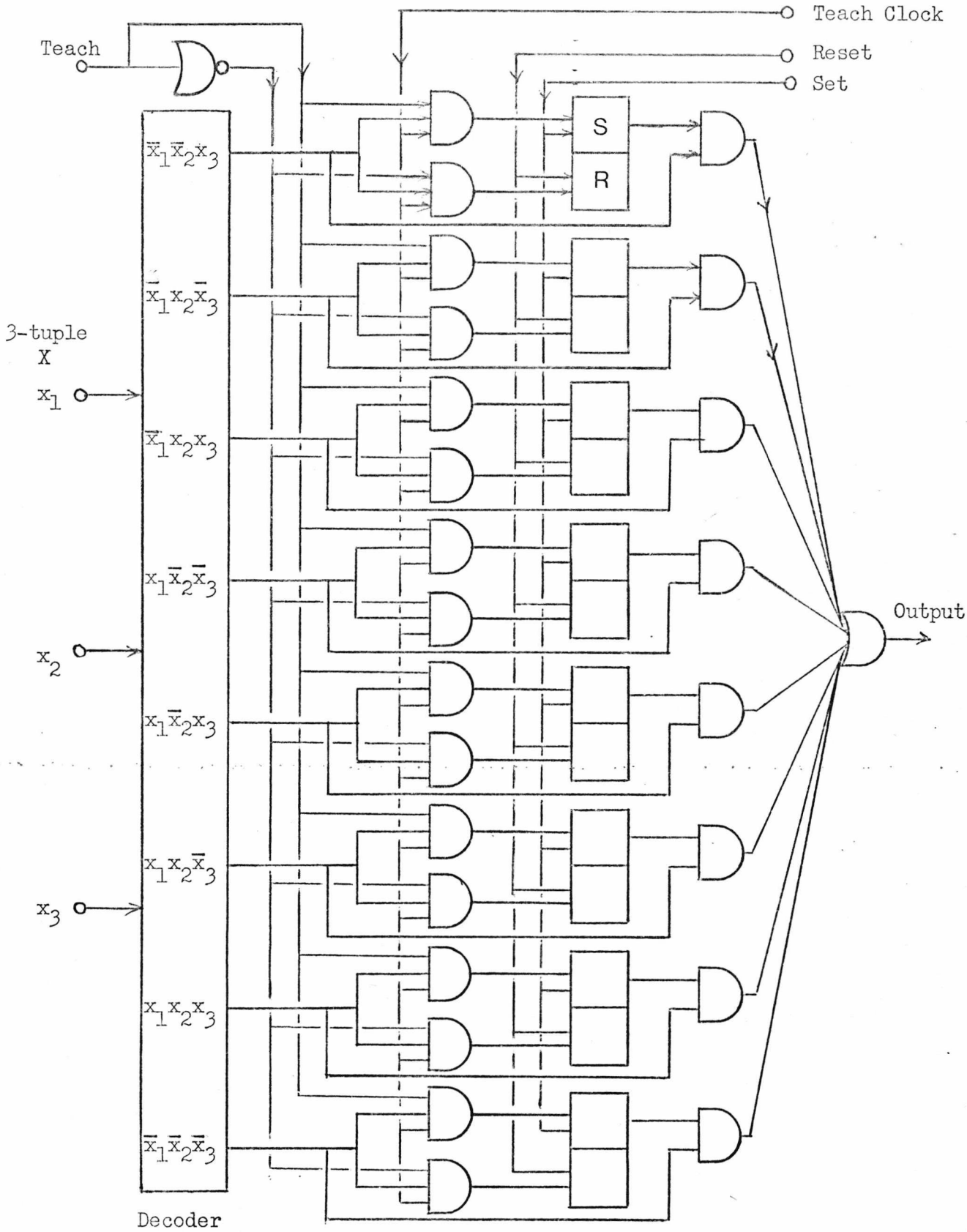

Fig. 3. 1. Logic Diagram of SLAM-8 Module.

at the output to all input configurations.

As an n-input SLAM module contains 2^n memory elements, one for each possible set of inputs, it is sometimes referred to as SLAM-2^n. Hence, a 3-input SLAM is sometimes called SLAM-8 and a 4-input SLAM as SLAM-16.

3.2 Mathematical Representation of a SLAM Function

The inputs to an n-input SLAM can be written as a vector $X = (x_1,.....,x_n)$, whose components can have only the logical values of '1' or '0'. The state of the memory elements can be represented by a vector $A = (a_1,.....,a_{2n})$, each of the components describing an individual memory element and is related to one of the possible values of the vector X. a_{2n} accounts for the case when all the components of X are '0'.

All the possible values of the vector X are mutually exclusive i.e. only one set of inputs can occur at a time.

$[X] = 1$ when the particular value occurs.

$[X] = 0$ otherwise.

The function of an n-input SLAM module can then be expressed in the form of a disjunctive logic expression.

$$g_s(X) = \sum_{i=1}^{2^n} a_i \cdot [X] \qquad (3.1)$$

Where a_i is equal to '1' when the particular memory element is set and equal to '0' when it is reset.

3.3 Similarity between a SLAM Module and a Φ-Discriminator

With an n-input SLAM, the maximum number of different input configurations is 2^n, the total number of distinct classifications of these 2^n sets of input into two classes (dichotomies) is 2^{2^n} and they can all be achieved by the SLAM module by appropriate settings of the memory elements (bits). Similarly with an nth. order Φ-discriminator with n inputs, any dichotomization can be achieved by adjusting the values of the weights. Like the n-input SLAM, which requires 2^n memory bits, an nth. order Φ-discriminator with n inputs needs 2^n weights (section 2.2.4).

The implementation of a 2nd order Φ-discriminator

which could classify binary patterns in 2 classes as a 2-input SLAM module will be shown and this can be extended to higher orders.

The discriminant function for a 2-input SLAM module, as from Eq. (3.1) can be written as

$$g_s(X)=a_1(\bar{x}_1x_2)+a_2(x_1\bar{x}_2)+a_3(x_1x_2)+a_4(\bar{x}_1\bar{x}_2) \qquad (3.2)$$

For this particular form, the substitution $\bar{x}_i=1-x_i$ can be made. Consequently, the Eq. (3.2) becomes

$$g_s(X)=a_1(1-x_1)x_2+a_2x_1(1-x_2)+a_3(x_1x_2)+a_4(1-x_1)(1-x_2)$$

$$=(a_2-a_4)x_1+(a_1-a_4)x_2+(a_4+a_3-a_1-a_2)x_1x_2+a_4 \qquad (3.3)$$

The discriminant function of a 2-input 2nd order Φ-discriminator from section 2.2.4, is

$$g(X)=w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4 \qquad (3.4)$$

Comparing equations (3.3) and (3.4), the weighting values for the Φ-discriminator to operate like the SLAM module can be made up from the state of the memory elements of the SLAM. In this case,

$$w_1 = (a_2 - a_4)$$

$$w_2 = (a_1 - a_4)$$

$$w_3 = (a_4 + a_3 - a_1 - a_2)$$

$$w_4 = a_4$$

Thus, if the state of the memory elements of a SLAM module is known, a Φ-discriminator can be implemented which would operate exactly as the SLAM module. But the converse is not possible because of the inherent generalization capability of a Φ-discriminator. A SLAM module on its own is not capable of generalization, its response to a set of inputs which has not been encountered before depends on the state of the related storage element prior to the training phase. The mode of usage of a SLAM module is different to that of a Φ-discriminator. In a Φ-machine, a Φ-discriminator would cover the whole retina, while in a PR using SLAMs (section 3.4) a bank of SLAMs are used to cover the retina, generalization is then inherently possible. Besides it is not realistic to have a single SLAM for the whole retina. To cover a 10x10 matrix by one SLAM module, the latter would require 2^{100} memory elements which is not feasible even by the present state of art in microminiaturization.

3.4 Basic SLAM-Pattern-Recognizer (SLAM-PR)

If there are R pattern classes, then the PR would consist of R SLAM-discriminators (connected to the same retinal points) followed by a maximum selector as illustrated in fig. 3.2. A SLAM-discriminator will be referred to as a discriminator in the following. The discriminators consist of the same number of SLAMs and are connected to the retina in the same way. The outputs of the SLAMs in each discriminator are fed into a summing device which adds all the positive responses. A positive response is a logical '1' output by a SLAM indicating the presence of a feature on which that particular SLAM has been trained.

During the recognition phase the maximum selector classifies the unlabelled pattern in the same class as the discriminator which gives the maximum response; in case of a tie between two or more discriminators, the maximum selector classifies the pattern into a reject class.

3.4.1 Training Phase

Initially all the SLAMs in the PR are reset, i.e. they would produce a negative response '0' to all sets

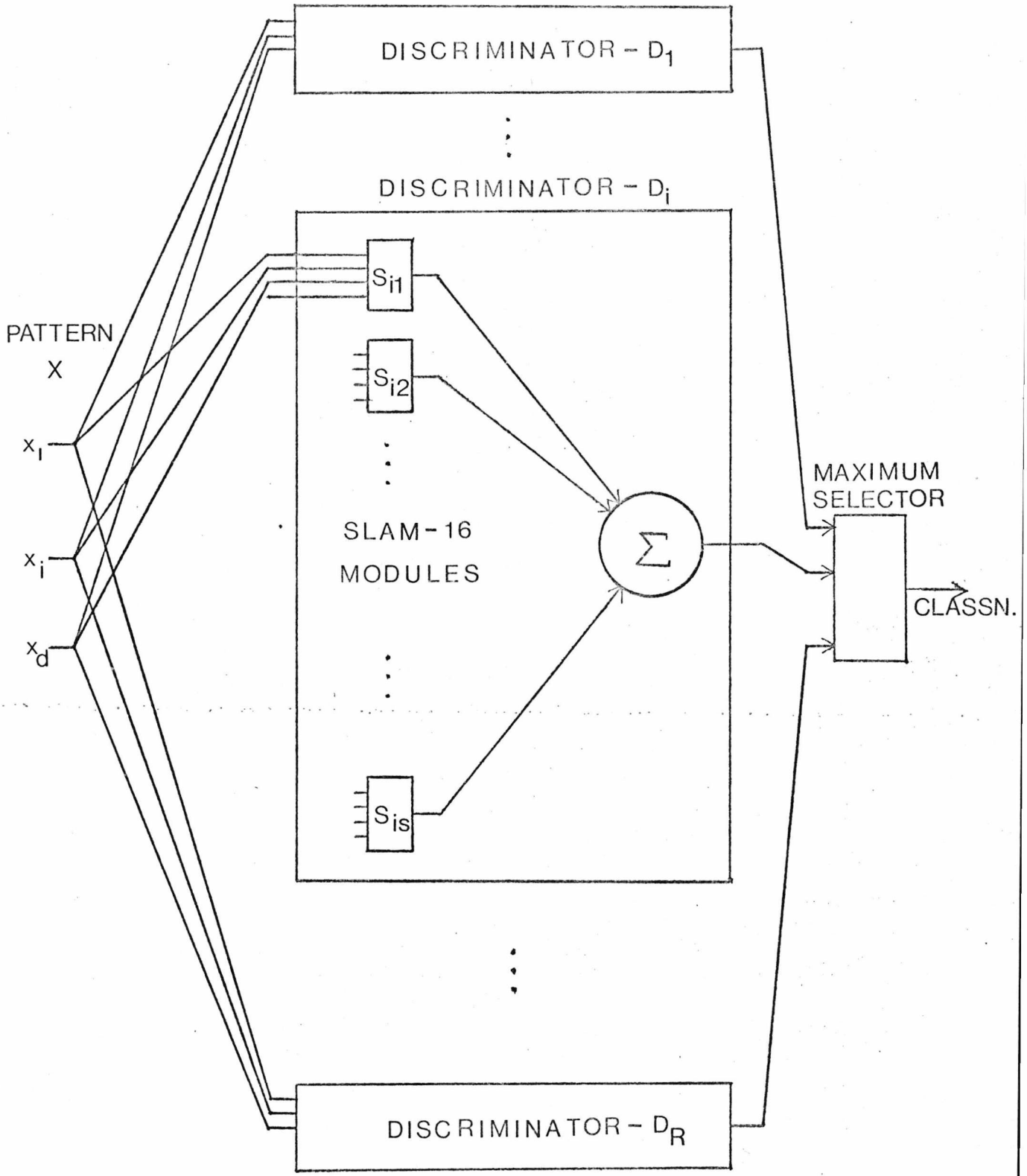

Fig. 3.2. Block Diagram of a Basic SLAM-PR.

of inputs. Each discriminator is then trained on its respective training patterns. The patterns from the training sets are presented one at a time to all the discriminators but only the SLAMs in the discriminator, which is related to the class of the pattern being presented, are taught. The inputs to the SLAMs in the discriminator are decoded and the memory bits which are addressed are set to '1' by the teaching strobe. If a bit is already set, it stays set. Setting memory bits to '1' is sometimes referred to in the following as '*filling*' the memory.

Each pattern from the training set is presented once only, and if the same pattern is presented during the *testing* phase, the SLAMs in the discriminator which has been trained on it, will all give a positive response.

3.4.2 Criterion for Training

Theoretically, for a discriminator consisting of n-input SLAM modules, it is possible to fill the memory completely with only 2^n patterns. The patterns can be devised in such a way that the SLAMs see a different set of inputs in each pattern. If the discriminator memory is filled completely, it will give the maximum output to all patterns and its discriminating power is nil. This is not desirable.

The response of a discriminator to a pattern of randomly distributed 0's and 1's is in fact a function of the percentage of the memory filled. If M is the overall percentage of the memory filled and assuming that the bits set are equally distributed among the SLAMs in the discriminator, a random pattern has an equal probability of presenting any of the possible states of the input n-tuples to a SLAM. The probability of a SLAM giving a positive response is therefore M% and the overall response of the discriminator would be approximately M% of the maximum response. A discriminator whose memory has more elements set is more likely to give a larger response to the same random pattern than a discriminator with fewer memory elements set.

Consequently the criterion for training a SLAM-PR in this thesis is based on all the discriminators being equally filled, whilst in most of the trainable PRs in literature, a measure of training is based on the number of patterns in the training set (Widrow & Hoff, 1960; Ullman & Kidd, 1969).

The patterns from different classes, because of their different degrees of variability, do not fill the memory at the same rate per pattern. The rate of filling the memory per pattern also depends on the average number of bits '1' per pattern for the class as will be discussed in the next section.

3.4.3 Variation in Rate of Filling the Discriminator Memory with Pattern Density

The density of a pattern is defined as the ratio of the pattern components which are '1' to the number of pattern components. The densities of typical patterns from different classes are expected to be different, e.g. between numerical characters '1' and '8', the density of the latter is likely to be greater because of the longer tracing. The dependence of the rate of filling the discriminator memory on the pattern density is best illustrated by examples, where the numbers of patterns, of different pattern densities, required to fill the memory are estimated.

In the following, the estimations are carried out for single SLAM modules, and are applicable to discriminators containing any number of SLAMs; if no two inputs of the same SLAM are connected to the same retinal point and the pattern components which are '1', are independently and randomly distributed in the pattern vector. The density of such a random pattern is then equivalent to the probability of bit '1' occurring at any retinal point and is written as $P_b(1)$.

The probability, $P_b(0)$, of bit '0' occurring is then

$$P_b(0) = 1-P_b(1)$$

Considering a SLAM-4, the number of different sets of inputs is 4. If $P_n(i)$, i=0,1,2,3, is the probability of a particular state (i) of a 2-tuple occurring,

$$P_n(0) = P_b(0)\ P_b(0)$$

$$P_n(1) = P_b(0)\ P_b(0)$$

$$P_n(2) = P_b(1)\ P_b(0)$$

$$P_n(3) = P_b(1)\ P_b(1)$$

If $Q_b(j)$, j=1,2,3,4, is the probability of j memory bits being set in the SLAM after it has been trained on j patterns of the same density,then

$$Q_b(1) = \sum_{j=0}^{3} P_n(j) = 1$$

$$Q_b(2) = \sum_{\substack{j=0 \\ k=0}}^{3} P_n(j)\ P_n(k)$$

$$Q_b(3) = \sum_{\substack{j=0 \\ k=0 \\ l=0}}^{3} P_n(j)\ P_n(k)\ P_n(l)$$

$$Q_b(4) = \sum_{\substack{j=0 \\ k=0 \\ l=0 \\ m=0}}^{3} P_n(j)\ P_n(k)\ P_n(l)\ P_n(m)$$

The estimated most likely number of patterns, $Q_p(j)$, required to set j bits in a SLAM-4 is therefore,

$$Q_p(j) = \frac{j}{Q_b(j)} \qquad j=1,2,3,4$$

The estimations for a SLAM-4 are carried out for different pattern densities (Appendix 1); arbitrary values of pattern density and also the average densities of the pattern classes from actual data patterns are chosen. The average density for numerals '1' is 0.217, for numerals '8' is 0.464 and the average for all hand-written numeric characters is 0.360 (see section 4.4). Similarly the estimations for SLAM-8 and SLAM-16 modules are done and the results are given in tables 3.1, 3.2, 3.3.

The relationship between the rate the memory is filled and the pattern density is symmetrical about the point when the pattern density is 0.5 because of the reciprocity of bits '1' and '0'; that is, the probabilities of bit '1' and bit '0' occurring can be interchanged in the above calculations. The estimations show that the highest rate of filling a SLAM or discriminator memory is when the pattern density is 0.5, i.e. when '1' and '0' are equally probable to occur at any retinal point; for lower (<0.5) or higher (>0.5) pattern density more patterns are required to fill the memory. Also for a given pattern density more patterns are required for larger SLAM modules.

For real data, however, the degree of variability

NUMBER OF BITS SET.	PATTERN DENSITY							
	0.1	0.2	0.3	0.4	0.5	0.217	0.464	0.360
1	1.0 E0	1.0 E0	1.0 E0	1.0 E0	1.0 E0	1.0 E0	1.0 E0	1.0 E0
2	6.1 E0	3.7 E0	3.0 E0	2.7 E0	2.7 E0	3.5 E0	2.7 E0	2.8 E0
3	6.2 E1	2.0 E1	1.1 E1	8.7 E0	8.0 E0	1.7 E1	8.1 E0	9.4 E0
4	2.5 E3	2.5 E2	8.6 E1	5.0 E1	4.3 E1	2.0 E2	4.4 E1	5.9 E1

Table 3.1. Estimated No. of Random Patterns to Fill a SLAM - 4 for various Pattern Densities.

0.217 - Average Pattern Density for Numerical Characters '1'.

0.464 - " " " " " " '8'.

0.360 - " " " " All Ten Numerical Characters.

NUMBER OF BITS SET.	PATTERN DENSITY							
	0.1	0.2	0.3	0.4	0.5	0.217	0.464	0.360
1	1.0 E0	1.0 E0	1.0 E0	1.0 E0	1.0 E0	1.0 E0	1.0 E0	1.0 E0
2	4.5 E0	2.9 E0	2.5 E0	2.3 E0	2.3 E0	2.8 E0	2.3 E0	2.4 E0
3	2.4 E1	8.9 E0	5.8 E0	4.8 E0	4.6 E0	8.1 E0	4.6 E0	5.1 E0
4	1.9 E2	3.4 E1	1.6 E1	1.1 E1	9.8 E0	2.8 E1	9.9 E0	1.2 E1
5	2.8 E3	1.8 E2	5.2 E1	2.9 E1	2.4 E1	1.4 E2	2.5 E1	3.5 E1
6	7.0 E4	1.4 E3	2.4 E2	1.0 E2	7.8 E1	9.7 E2	8.1 E1	1.3 E2
7	3.6 E6	2.0 E4	1.7 E3	5.3 E2	3.6 E2	1.2 E4	3.8 E2	7.6 E2
8	7.0 E8	7.0 E5	2.7 E4	5.4 E3	3.3 E3	3.4 E5	3.5 E3	8.9 E3

Table 3.2 Estimated No. of Random Patterns to Fill a SLAM - 8 for various Pattern Densities.

NUMBER OF BITS SET.	PATTERN DENSITY							
	0.1	0.2	0.3	0.4	0.5	0.217	0.464	0.360
1	1.0 E00	1.0 E00	1.0 E00	1.0 E00	1.0 E00	1.0 E00	1.0 E00	1.0 E00
2	3.6 E00	2.5 E00	2.3 E00	2.2 E00	2.1 E00	2.5 E00	2.1 E00	2.2 E00
3	1.4 E01	5.9 E00	4.3 E00	3.8 E00	3.7 E00	5.5 E00	3.7 E00	3.9 E00
4	6.7 E01	1.5 E01	8.2 E00	6.4 E00	6.0 E00	1.3 E01	6.1 E00	6.9 E00
5	4.1 E02	4.3 E01	1.7 E01	1.1 E01	1.0 E01	3.4 E01	1.0 E01	1.3 E01
6	3.3 E03	1.4 E02	3.7 E01	2.1 E01	1.7 E01	1.0 E02	1.8 E01	2.5 E01
7	3.5 E04	5.6 E02	9.2 E01	4.1 E01	3.3 E01	3.7 E02	3.4 E01	5.2 E01
8	5.0 E05	2.6 E03	2.6 E02	9.0 E01	6.6 E01	1.6 E03	6.9 E01	1.2 E02
9	9.6 E06	1.5 E04	8.5 E02	2.2 E02	1.5 E02	7.9 E03	1.6 E02	3.3 E02
10	2.5 E08	1.1 E05	3.2 E03	6.2 E02	3.8 E02	5.0 E04	4.0 E02	1.0 E03
11	9.5 E09	1.0 E06	1.5 E04	2.0 E03	1.1 E03	3.9 E05	1.2 E03	3.7 E03
12	5.4 E11	1.2 E07	8.6 E04	8.0 E03	3.9 E03	4.0 E06	4.2 E03	1.7 E04
13	4.9 E13	2.1 E08	6.5 E05	3.9 E04	1.7 E04	5.7 E07	1.9 E04	9.3 E04
14	7.3 E15	5.2 E09	6.7 E06	2.6 E05	9.6 E04	1.2 E09	1.1 E05	7.1 E05
15	2.2 E18	2.2 E11	1.1 E08	2.6 E06	8.3 E05	4.1 E10	9.6 E05	8.2 E06
16	2.2 E21	2.2 E13	3.7 E09	5.2 E07	1.4 E07	3.3 E12	1.7 E07	1.9 E08

Table 3.3. Estimated No. of Random Patterns to Fill a SLAM-16 for various Pattern Densities.

between the patterns from the same class and the constraint imposed upon the distribution of bits '1' by the characteristics of the patterns have to be considered. That is, the joint probabilities of occurrence of features come into play and generally increase the number of patterns required to fill the memory. On the other hand, a high degree of variability would reduce the number of patterns needed to fill the memory (section 4.4).

3.4.4 Testing Phase

To estimate the performance of the SLAM-PR after training, the system is tested on test patterns. These are patterns of known classification and although they are drawn from the same populations as the training patterns, some differences must be expected.

Each pattern from the test data is presented to all the discriminators and every SLAM will respond positively if the state of the n-tuple it 'sees' has been encountered during the training phase. The outputs of the SLAMs for each discriminator are added and the maximum selector classifies the pattern accordingly. There are 3 possible classifications: a correct classification which is referred to as recognition, a rejection, or a misclassification which is referred to in the following as a

substitution error.

As the patterns in the test data for each class are assumed to be only representatives of patterns of the class and are not the complete repertoire of the possible patterns for the class, the performance based on the test data is only a statistical estimate.

CHAPTER 4

SIMULATIONS OF SLAM-PATTERN-RECOGNIZERS

4.1 Configuration of the Computer System

All the experiments in this thesis have been simulated on a general purpose computer, fig. 4.1. It is a Honeywell DDP-516 with a core memory of 8K words, backed by a magnetic tape transport. The central processor handles 16-bit words in parallel. The memory read/write cycle time is 0.96 microsecond and the majority of the 72 instruction set are completed in one or two cycles.

For the simulations, the assembly language for the machine, DAP, has been used throughout for economy of storage and processing time. The data for the simulation is stored on magnetic tape and the results are output on the teletype.

4.2 Experimental Data and Preprocessing

All the experiments in this report use the same data, which consists of 400 examples of each of the 10 numerals. They are taken at random from handwritten addresses on envelopes in the mail and digitized by

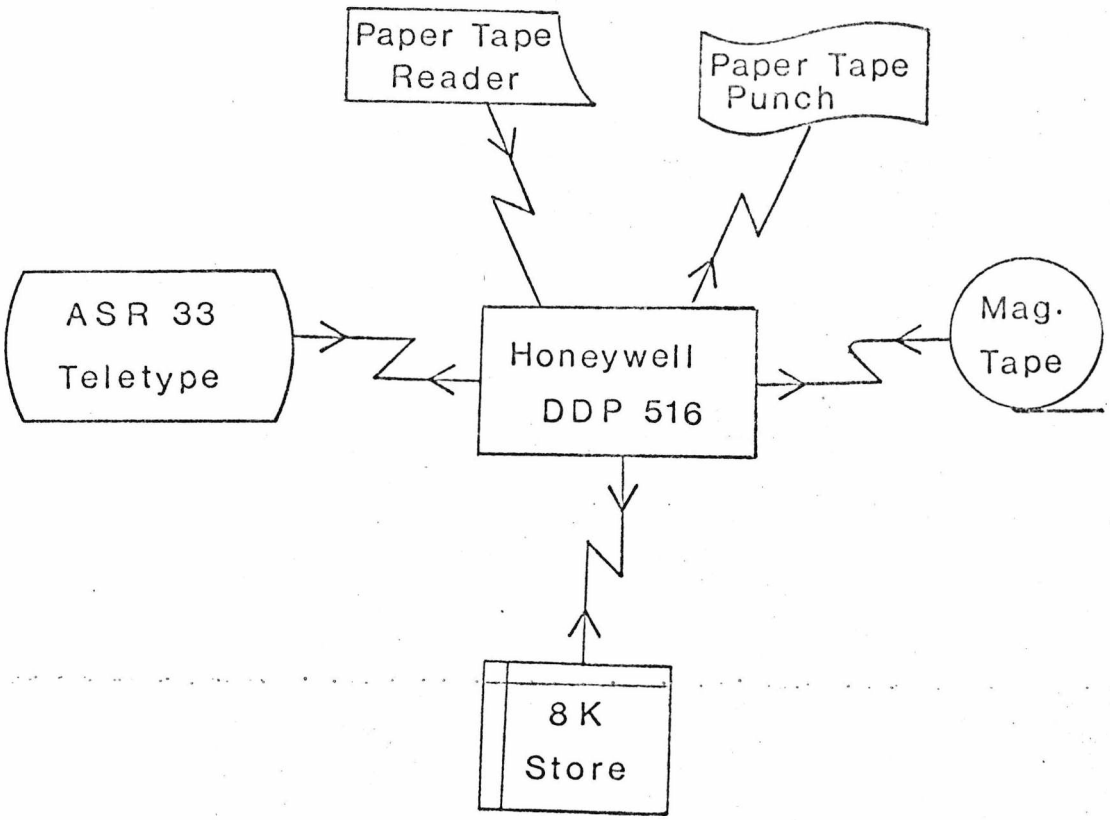

Fig. 4.1. Configuration of the General Purpose Computer Used for the Simulations.

means of a flying spot scanner into 22x30 binary patterns*. The scanner does not have automatic centering or size-normalizing facilities, the characters are roughly 'top-left-adjusted' by a human operator and there is no normalization of height to width ratio or of orientation. In some cases the digitization is imperfect, in that part of the character is missing in the 22x30 array. Imperfect segmentation is due to the handwritten characters being too large for the 'digitizing' frame or due to incorrect positioning of the frame.

Owing to the large number of patterns used in the following simulations and the restricted core memory, the 22x30 patterns have been reduced to more manageable 12x15 patterns. This is achieved by adding an extra blank column on each side of the original pattern giving an 24x30 array and reducing the size by dividing the whole pattern into a set of rectangular 2x2 sub-matrices. At each submatrix, a binary output is generated by an 4-input 'or' gate, giving a 12x15

---

*The data was made available through the courtesy of P.O. Research Station at Dollis Hill, London.

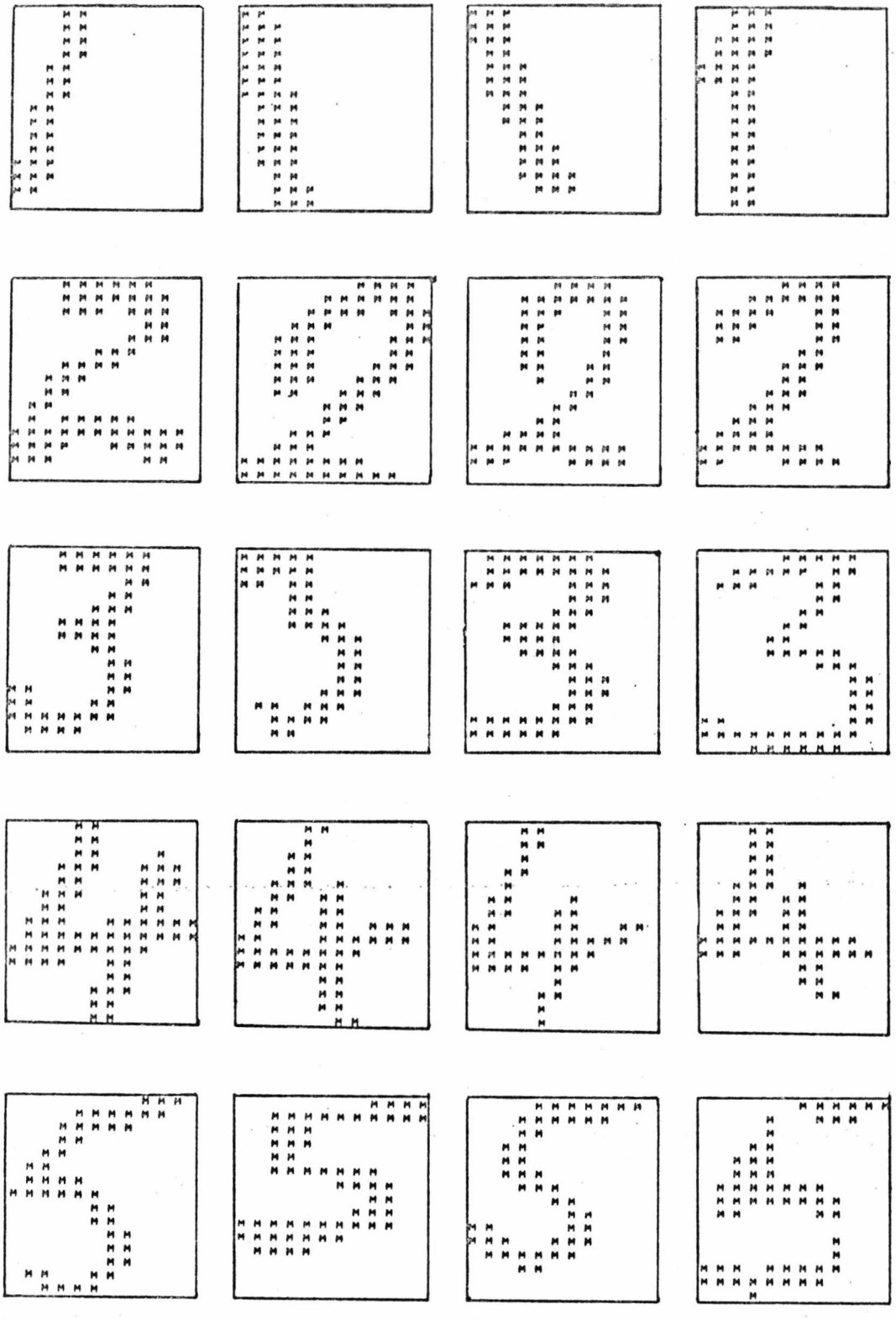

Fig. 4.2a. Typical Examples of Reduced Patterns.

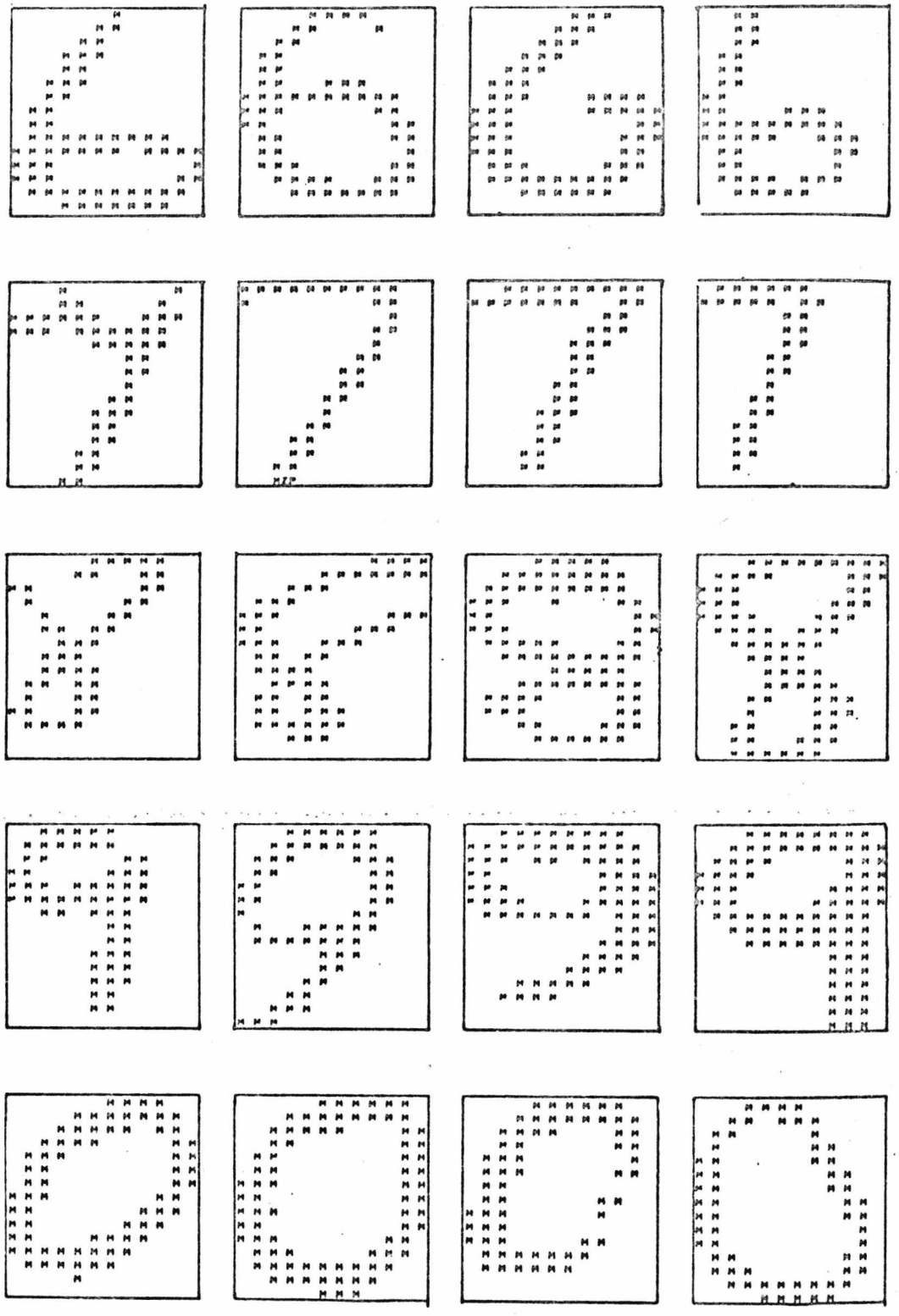

Fig. 4.2b. Typical Examples of Reduced Patterns.

pattern.

One result of the reduction process is that some narrow loops in numerals such as '6' or '8' or '9' may be filled up, and the stroke becomes thicker. The resulting patterns however are quite acceptable as shown by the typical examples in figs. 4.2a and 4.2b.

The only preprocessing all the patterns are subjected to, is the elimination of single-isolated bits: a bit '1' surrounded by bits '0' or a bit '0' surrounded by bits '1'. It is a non-reversible process and it is assumed that completely isolated bits are due to noise and no information is lost by their removals. A pattern, which is now expressed by a 12x15 array, is scanned by a 3x3 submatrix every bit horizontally and vertically. For retinal points which lie on the edges, a dummy row or column of bits '0' is added all round the pattern.

a	b	c
d	x	e
f	g	h

FIGURE 4.3

Retinal Points Involved in Elimination of Isolated Bits

Fig. 4.3 shows a retinal point x, and the eight points around it. Each point is treated as a binary variable and x is changed to f_1, if and only if

$$a = b = c = d = e = f = g = h = f_1$$

That is, the value of x is changed or left equal to f_1 irrespective of its value if all the bits around it are equal. The values of a,b,...,h are the values from the un-preprocessed pattern. The change of state of one retinal point does not affect the determination of f_1 for the neighbouring points and the preprocessing for all the components of a pattern can be carried out, in hardware, in parallel.

The resulting preprocessed patterns are used as data patterns for all the simulations in this work.

4.3 Size of SLAM Module

Both SLAM-8 and SLAM-16 modules have already been manufactured as M.O.S.T. devices (Albrow *et al.*, 1967; Aleksander & Glover, 1970), and they can be used as building blocks for larger modules. It should be noted that random-access-memory (RAM) microcircuits can

also be used as SLAM modules, for instance a 256 bit RAM could represent 16 SLAM-16 or 32 SLAM-8 modules.

Ullmann (1969) has found that for a given number of training patterns, the performance of a SLAM-PR improves to an optimum and then declines with increasing size of SLAMs (will be discussed in chapter 6). For the following simulations of SLAM-PR (fig. 3.2), SLAM-16 modules have been chosen for the following reasons:

(i) A 16-bit word of the DDP-516 computer could conveniently represent the 16 memory elements of each SLAM. This enables the maximum utilization of the core storage memory and demands less programming effort, hence economizes on the core memory for the simulation programs.

(ii) The probability of complete (i.e. fruitless) filling of a SLAM module decreases with the size of the module. This provides an argument for choosing the largest possible size of SLAM module, provided that there is enough data, the simulation time is not excessive and the computer facilities permit it.

(iii) However, an argument for not making the SLAM too large is obtained by extrapolating values from tables 3.1, 3.2 and 3.3. It is estimated that the 400 patterns available for each class would on the average half-fill the memory of a SLAM-32 discriminator. With the given data it would not be possible to study PRs, consisting of SLAM-32 modules, for levels of memory filled beyond half-way, especially if a certain number of patterns are reserved as testing patterns.

For all the simulations of SLAM networks here, SLAM-16 modules are used, and the size of a SLAM-PR is defined as the number of SLAM-16 modules per discriminator.

4.4 Size of Training Set

The size of the training set has been chosen empirically. The percentage of the memory filled for a discriminator, containing 180 SLAMs, with the number of patterns is determined for each class separately.

The results are shown in fig. 4.4. As the SLAMs are randomly connected, the number of SLAMs in the discriminator should not have any bearing on the results but a large number of SLAMs are used to smooth the statistical variations.

As expected (section 3.4.3) the memory becomes increasingly difficult to fill with increasing pattern numbers, and there is some correlation between the rate at which the memory is filled and the pattern density. The average pattern density for the patterns in each class is given in table 4.1. Numerals '1', with the lowest average pattern density, have the lowest rate of filling the memory. While numerals '8', with the highest average pattern density, have the highest rate. In between the two extremes, there are some discrepancies, for instance, numerals '0', though with a higher average pattern density, do not fill as much memory as numerals '4'. This can be attributed to the fact that numerals '0' have a relatively lower degree of variability.

The maximum size of the training set for each class is limited to 100 patterns, as for each class the percentage of memory filled tends to level off (fig. 4.4). In all the following experiments, the same patterns are used in training, leaving 300 patterns for each class for estimating the performance

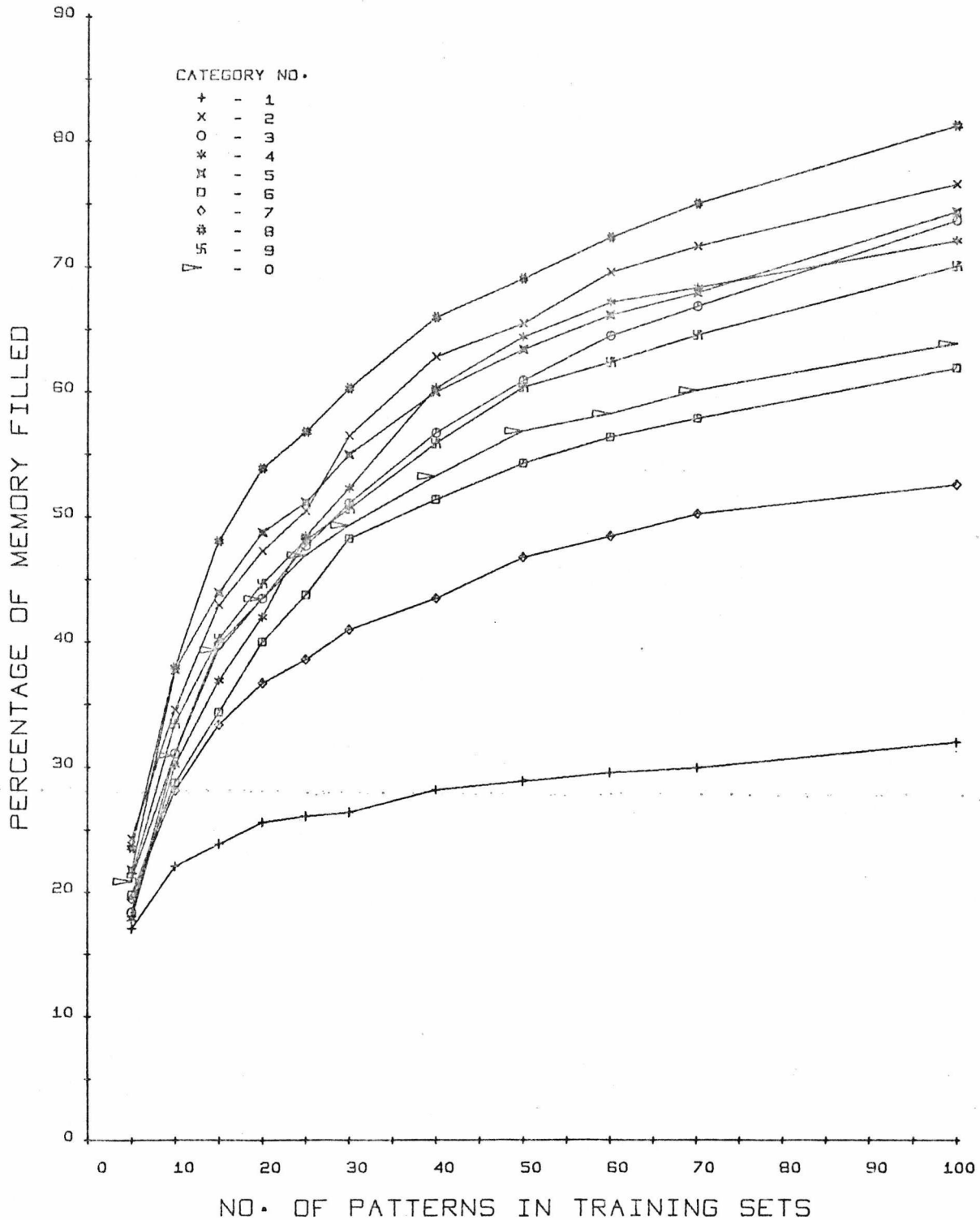

Fig. 4.4. Memory Filled versus No. of Training Patterns.

Pattern Class (Numeral)	Average Pattern Density
1	0.217
7	0.291
4	0.356
5	0.367
6	0.368
3	0.376
9	0.377
2	0.379
0	0.398
8	0.464

Table 4.1. Average Pattern Densities Arranged in Ascending Order of Magnitude.

of the systems in the testing phase.

4.5 Basic SLAM Pattern Recognizer

PRs with a single layer of SLAMs as shown in fig. 3.2 are simulated to find how the number of SLAMs per discriminator would affect the performance of the system. Concomitantly, the effect of different levels of memory filled on the performance is observed. Ullmann (1969) reports similar experiments, but he uses the size of the training sets as a measure for training, while here the criterion for training is the percentage of memory filled.

SLAM-PRs with 10,20,....,160 SLAMs per discriminator are simulated. In the absence of any algorithm to select n-tuples optimally, the SLAMs are randomly connected to the retina. The major operations in the simulations are shown in fig. 4.5. The connections of the SLAMs are generated by a programmed-in pseudo-random number generator (appendix 2). The only constraint imposed on the distribution of the connections is that no two inputs to the same SLAM are connected to the same retinal point. The 10 discriminators, one for each pattern class, are equally connected.

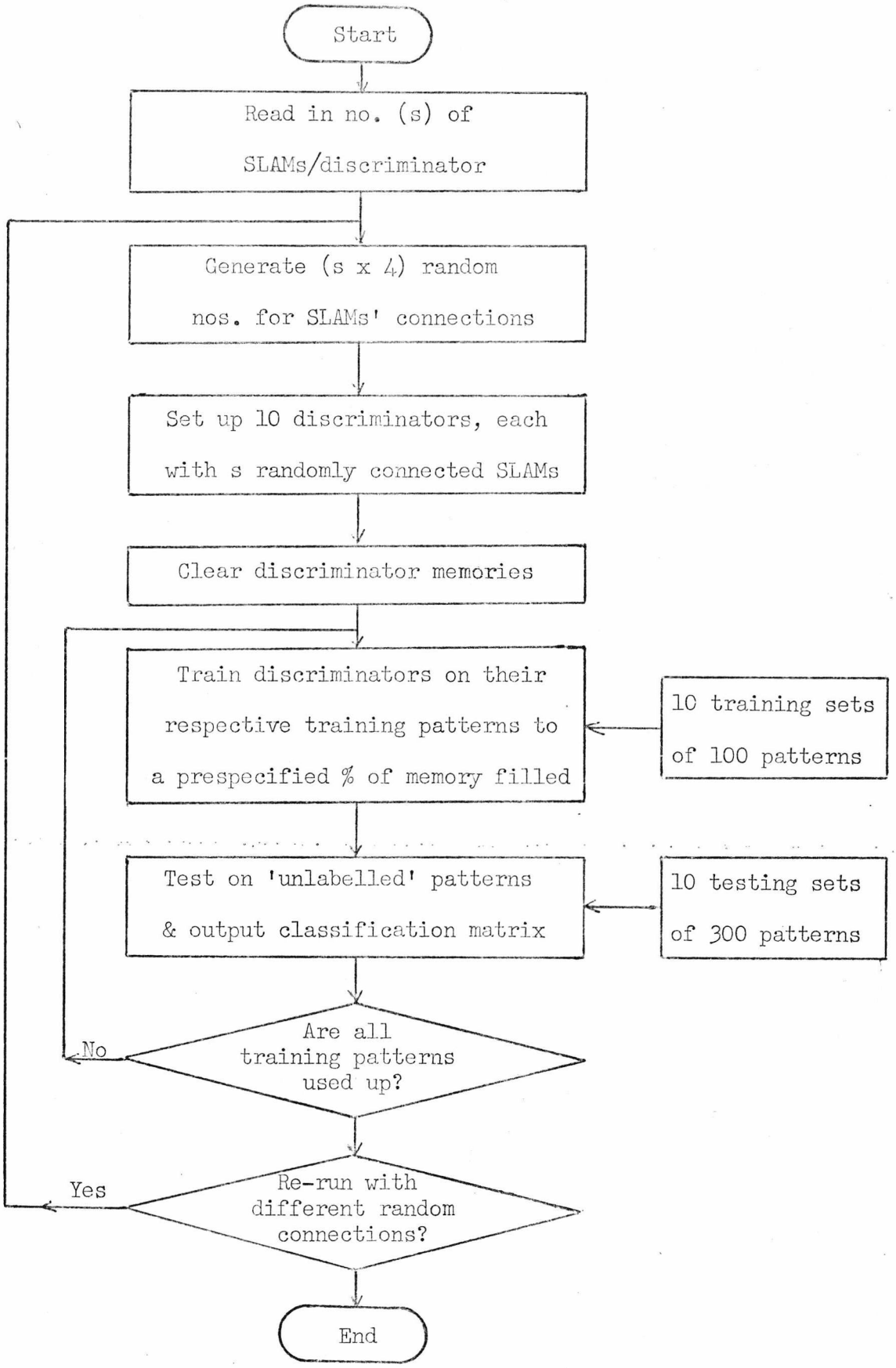

Fig. 4.5. Major Operations in SLAM-PR Simulation.

During the training phase, it is not always possible to fill the discriminators equally as the training is done on integral pattern numbers and the number of memory bits set by each pattern is not necessarily equal. The discriminators are, therefore, filled as equally as possible.

The training is stopped when the level of memory filled reaches or just overshoots prespecified values and the performance of the PR is estimated. The resulting classifications of the testing patterns are printed out as a classification matrix, an example of which is shown in fig. 4.6. The classification matrix indicates how the 300 testing patterns from each class are classified, whether correctly, incorrectly (substitution error), or rejected.

Five runs are made for each PR size, each with a different set of random connections for the SLAMs to the retina. The mean results, and the mean levels of memory filled are taken. The standard deviations for the recognition rate and the level of memory filled are tabulated in fig. 4.7a and plotted in fig. 4.7b. The latter shows a slight correlation between them.

The standard deviation for the recognition rate is already normalized as each system is tested on the same number of patterns. On the other hand, the

	CLASSIFICATION										REJECTION	SUBSTITUTION
DISCRIMINATOR	1	2	3	4	5	6	7	8	9	0		
NO. OF TRAINING PATTERNS	30	6	8	8	6	8	9	6	6	7		
% OF MEMORY FILLED	25.0	26.9	25.4	27.0	26.0	26.8	26.2	26.3	25.9	26.5		
ACTUAL PATTERN CLASS												
1	299	0	0	0	0	0	1	0	0	0	0	1
2	34	181	11	4	1	2	48	3	3	6	7	112
3	7	20	215	0	6	1	31	0	2	12	6	79
4	13	5	4	213	0	27	2	0	29	2	5	82
5	31	0	65	9	133	22	5	1	13	11	10	157
6	33	0	5	5	4	241	0	0	1	5	6	53
7	20	0	0	0	0	0	272	0	3	3	2	26
8	14	12	39	6	32	14	6	124	36	7	10	166
9	9	1	2	8	5	0	15	2	249	4	5	46
0	3	0	1	1	0	7	7	0	8	271	2	27

Average Memory Filled: 26.2%
Overall Recognition Rate: 73.3%
" Rejection " : 25.0%
" Substitution " : 1.8%

Fig. 4.6. A Typical Classification Matrix, Showing How the 300 Testing Patterns from Each Class are Classified. The Number of Patterns Used for Training and the % of Memory Filled for Each Discriminator Are Also Printed Out.

No. of SLAMs per Discriminator	Normalized Standard Deviation for	
	Recognition Rate	Level of Memory Filled
10	1.31	3.26
20	1.26	1.98
40	0.95	2.78
60	0.80	2.04
80	0.57	1.31
100	0.56	0.85
120	0.76	1.44
140	0.63	1.14
160	0.35	0.68

Fig. 4.7a. Standard Deviation of Recognition Rate & Memory Filled.

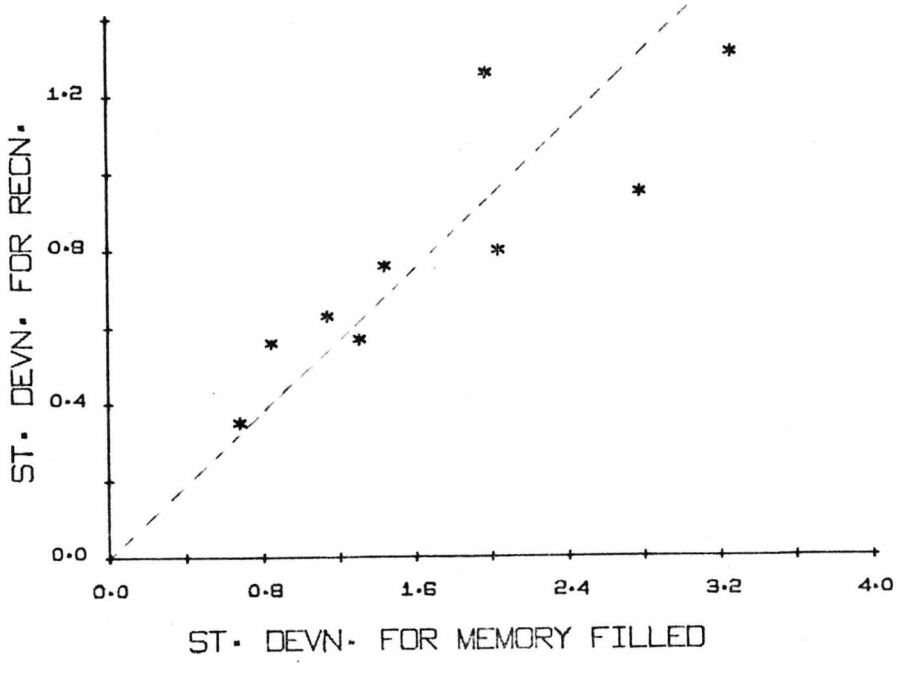

Fig. 4.7b. Standard Deviation of Recognition Rate versus Standard Deviation of Memory Filled.

standard deviation for the level of memory filled has to be normalized because of the different number of memory elements in different PR sizes. For example, a SLAM-PR using 10 SLAMs per discriminator has a total number of (10x10x16) memory elements, whereas one using 20 SLAMs per discriminator has (20x10x16) memory elements. In figs. 4.7a and 4.7b, the standard deviations are all normalized to the number of memory bits in a SLAM-PR containing 10 SLAMs per discriminator.

The means results for the various SLAM-PR sizes (10, 20,..., 160 SLAMs/discriminator) are plotted.

Figs. 4.8a and 4.9a show the recognition rate for the different systems and at different levels of memory filled, but plotted differently to illustrate better the characteristics of a SLAM-PR. The effect of levels of memory filled on the performance is more obvious in the former, while the latter shows how an increase in the number of SLAMs affect the recognition rate. Similarly, the effects on the rate of misclassification can be seen from figs. 4.8b and 4.9b, and the effects on the rejection rate can be seen from figs. 4.8c and 4.9c.

Fig. 4.8a is a series of recognition rate curves versus percentage of memory filled, for the different

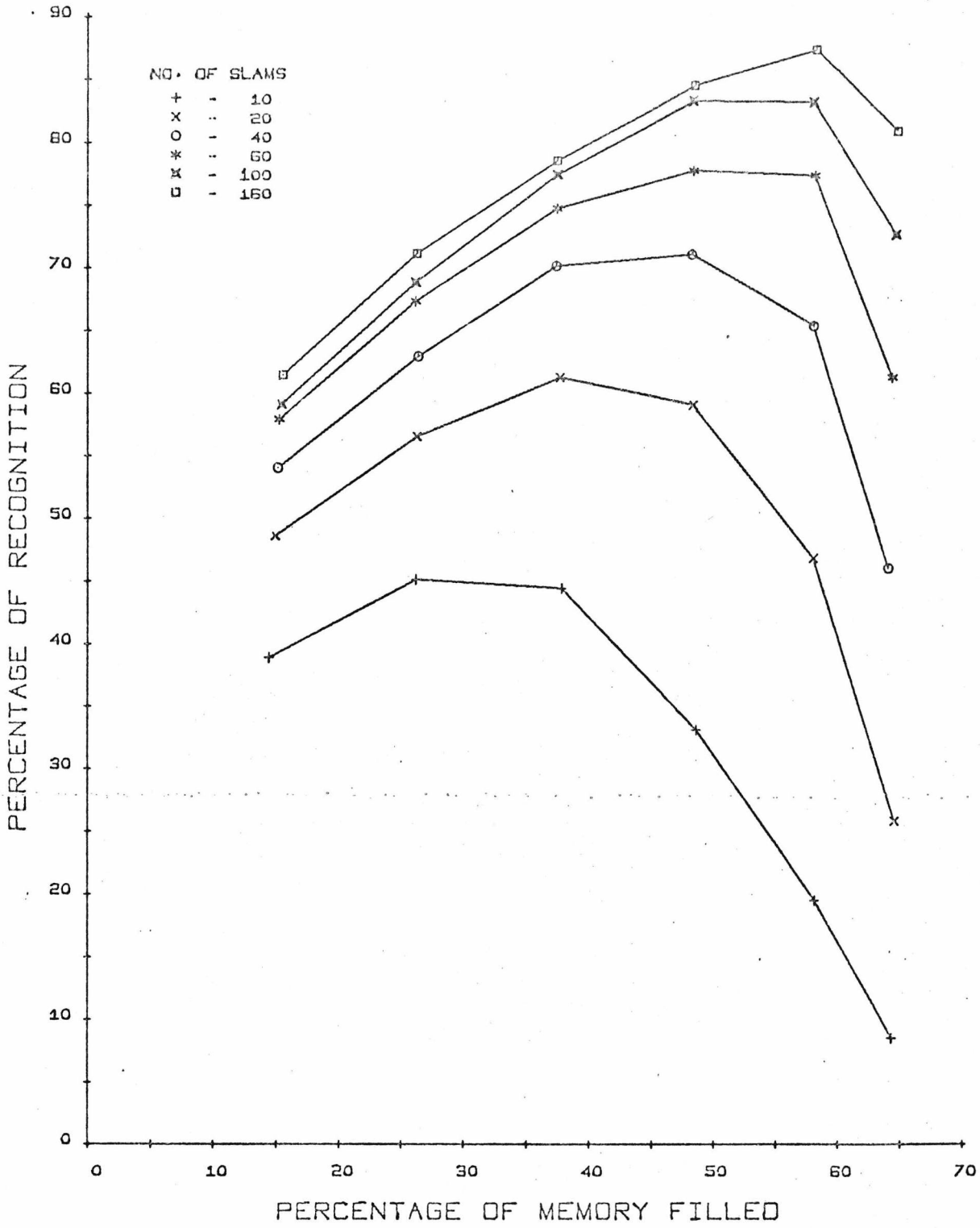

Fig. 4.8a. Recognition Rate versus Memory Filled.

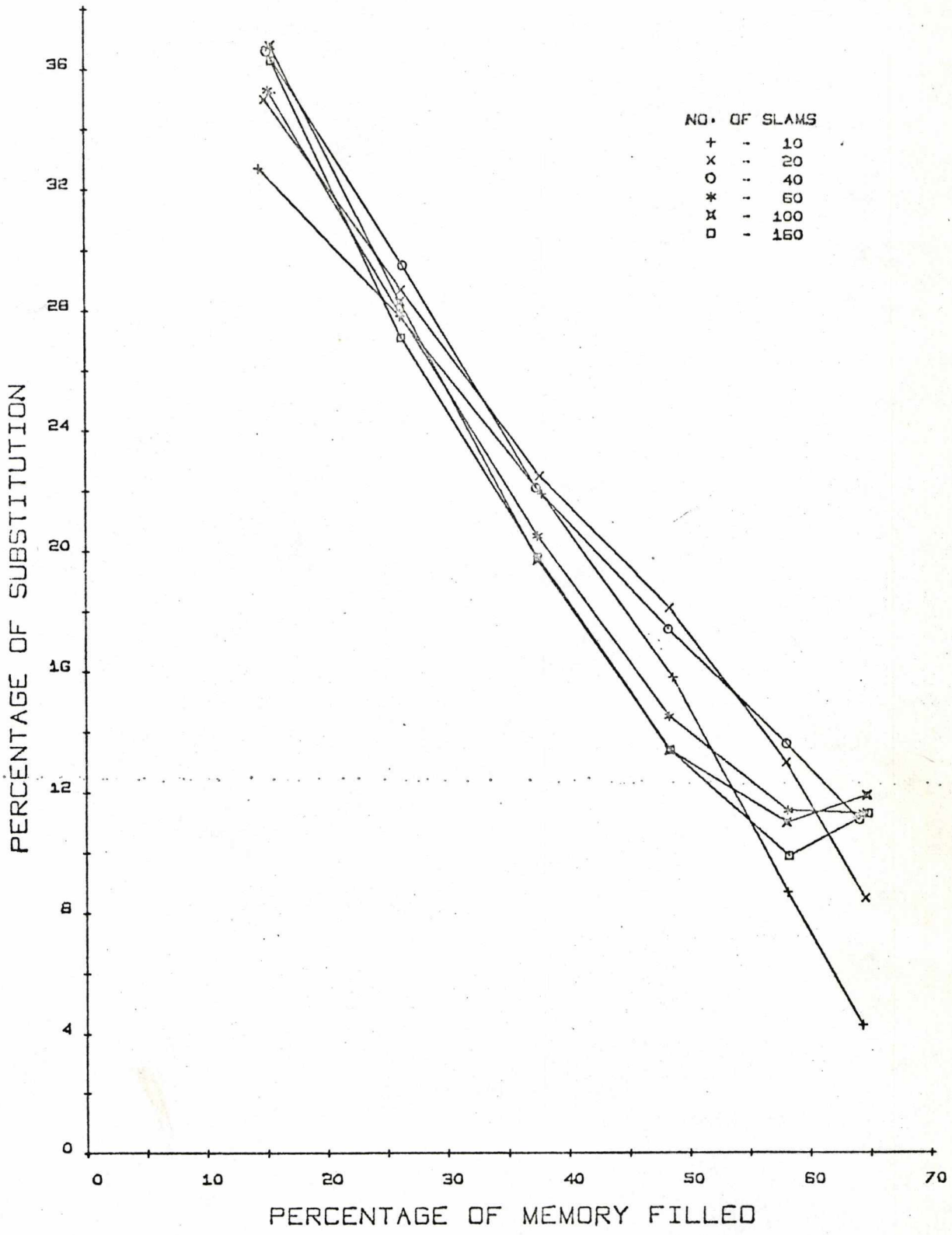

Fig. 4.8b. Substitution Rate versus Memory Filled.

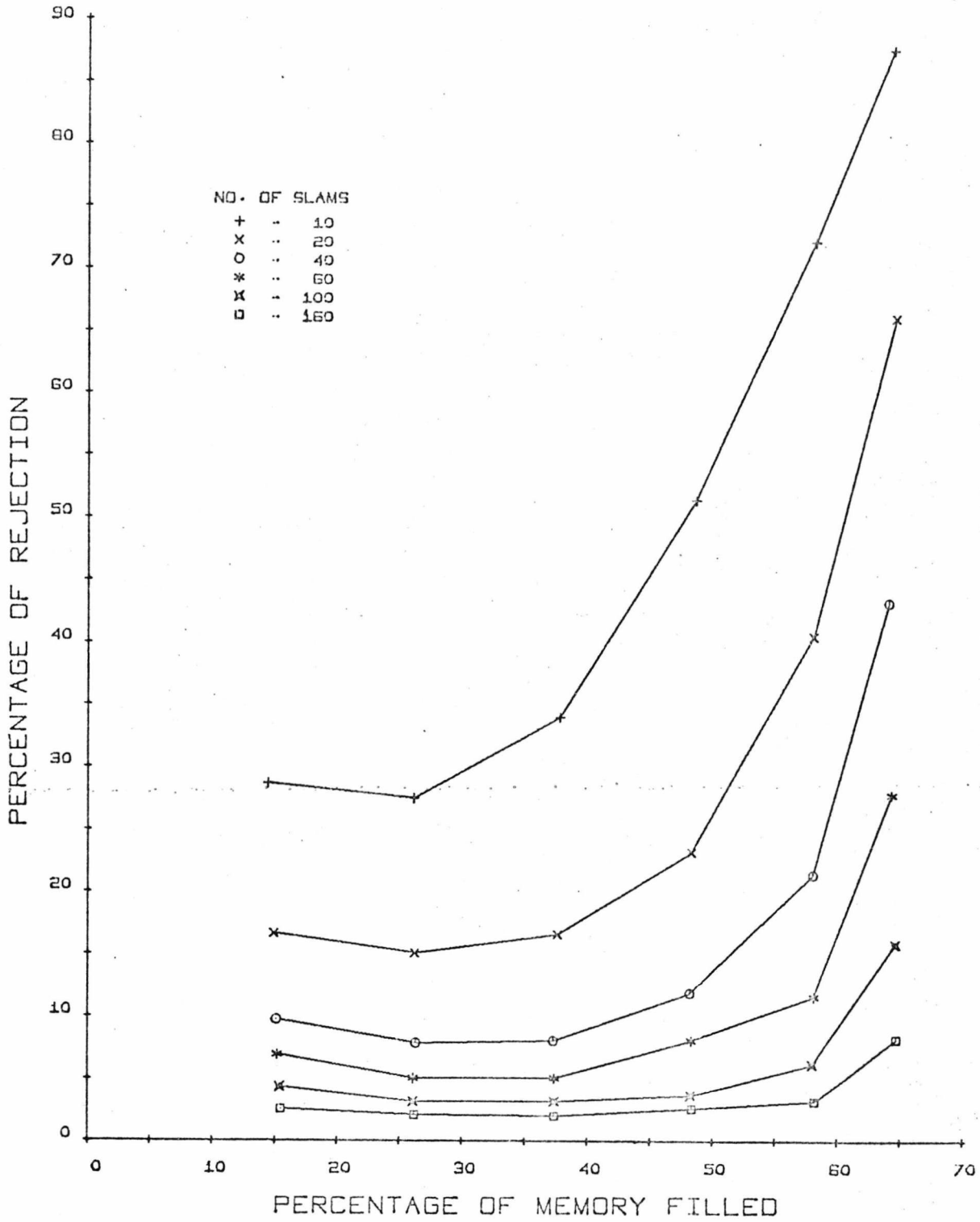

Fig. 4.8c. Rejection Rate versus Memory Filled.

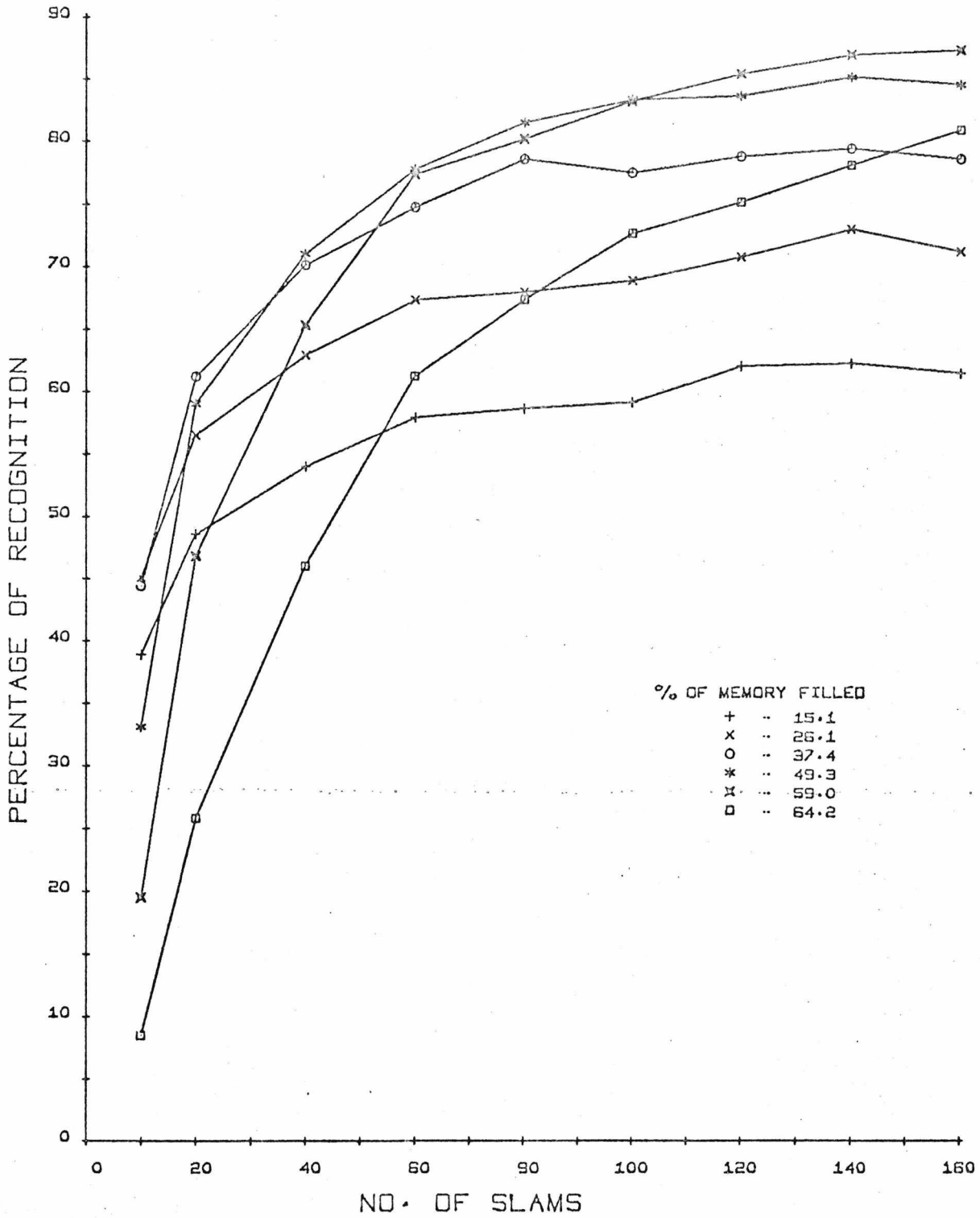

Fig. 4.9a. Recognition Rate versus No. of SLAMs.

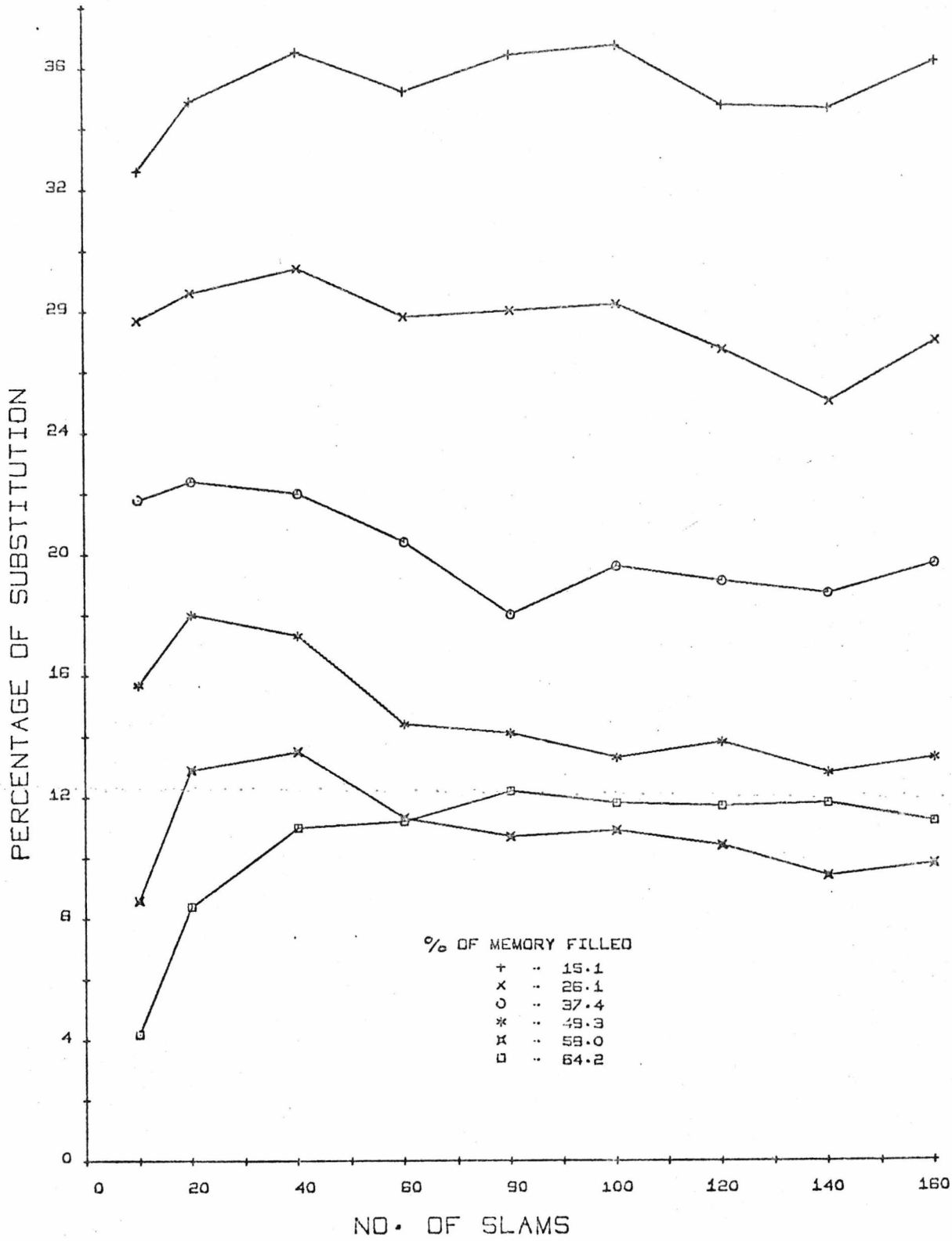

Fig. 4.9b. Substitution Rate versus No. of SLAMs.

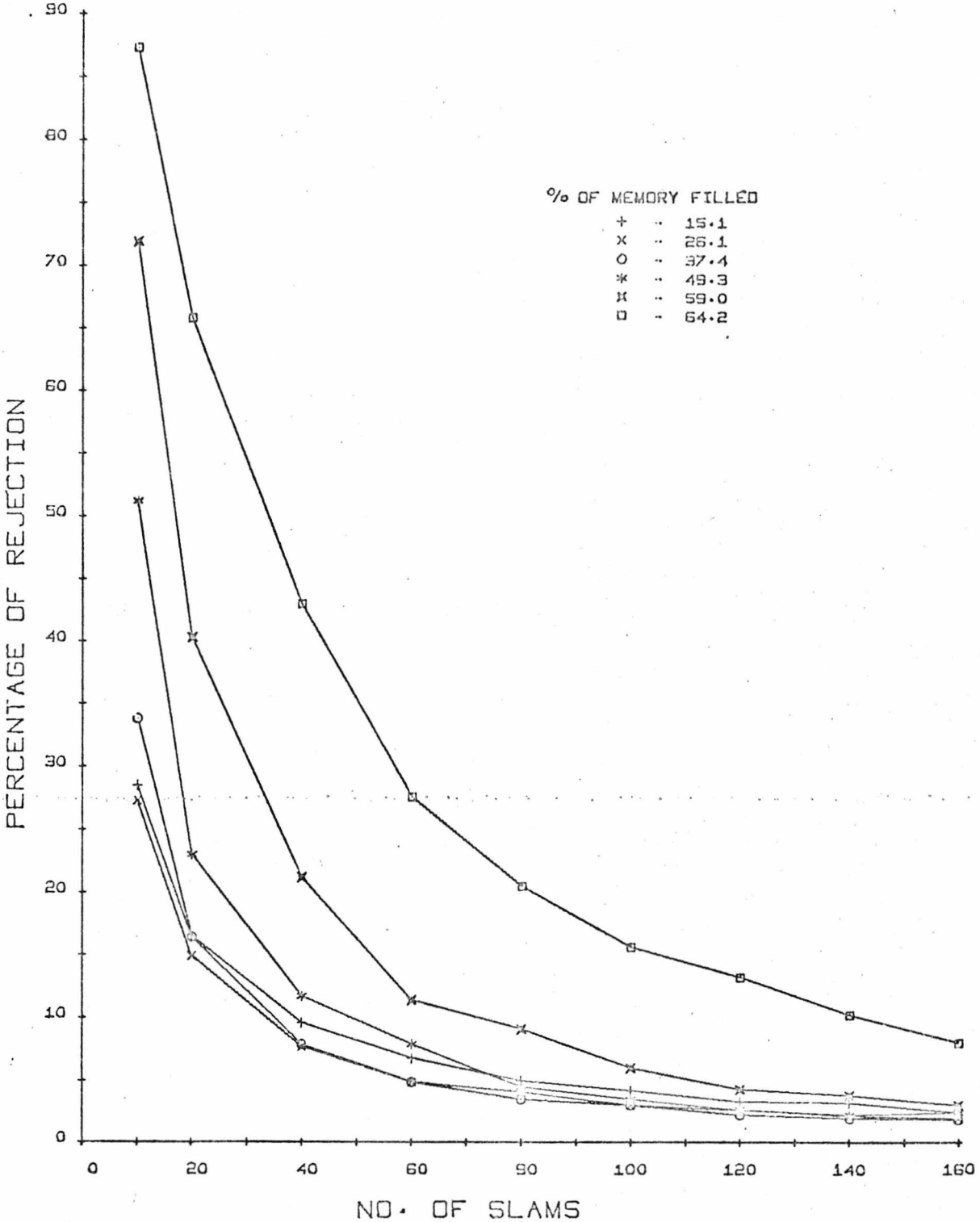

Fig. 4.9c. Rejection Rate versus No. of SLAMs.

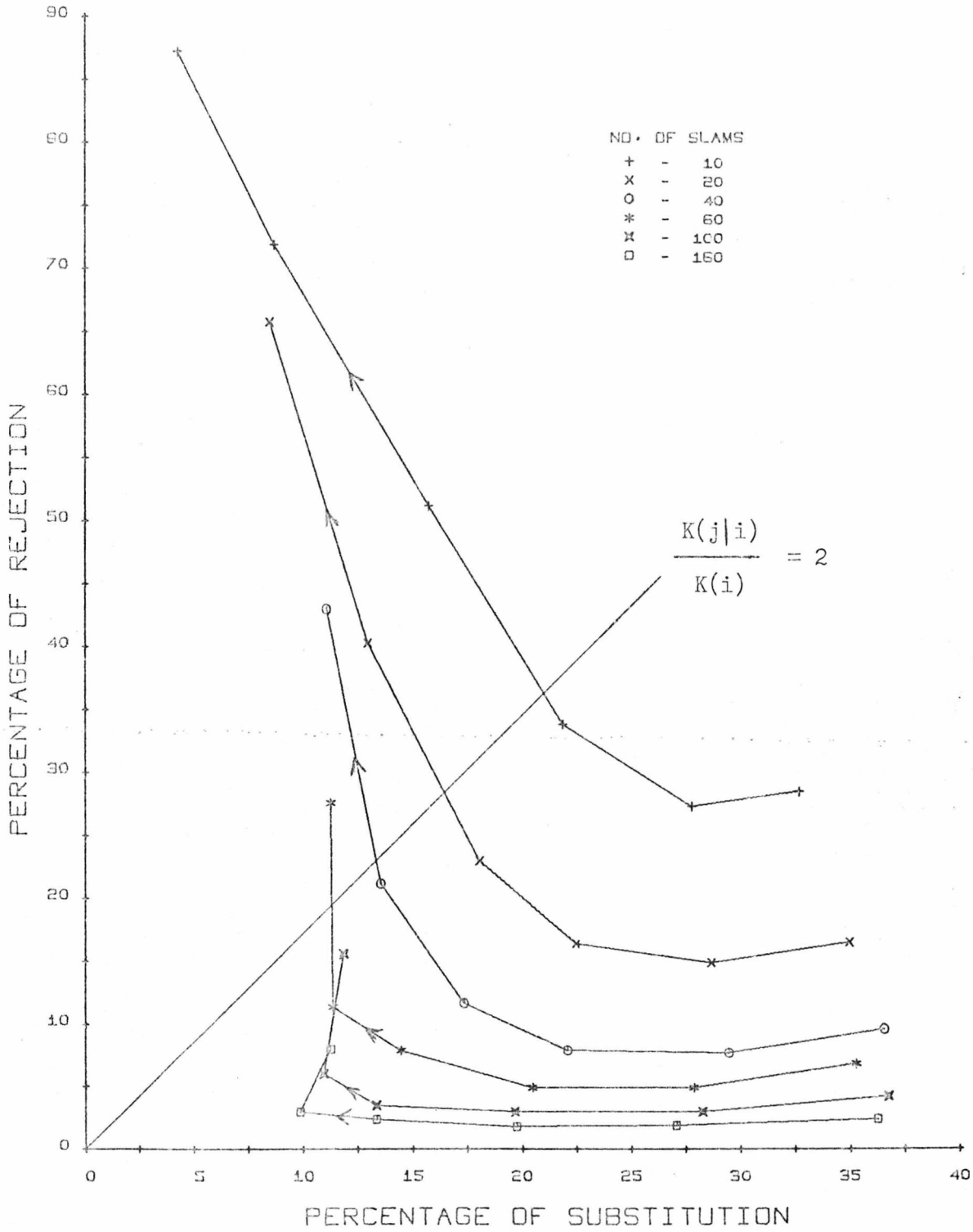

Fig. 4.10. Rejection Rate versus Substitution Rate.

sizes of SLAM-PR. It shows that for all the systems, the recognition rate reaches an optimum and then decreases as more memory is filled.

Fig. 4.8b is a series of substitution rate plots against percentage of memory filled, for the various systems. It shows that substitution rate decreases with memory filled.

Fig. 4.8c illustrates the variation of rejection rate with levels of memory filled. It shows that rejection rate increases with memory filled for all the systems.

Fig. 4.9a is a series of recognition rate curves against number of SLAMs for various levels of memory filled. It shows that for a certain level of memory filled, the recognition rate increases but tends to level off with increasing number of SLAMs.

Fig. 4.9b shows that for a given level of memory filled, the substitution rate remains more or less constant with the number of SLAMs.

Fig. 4.9c illustrates that as the number of SLAMs increases the number of rejections decreases irrespective of the level of memory filled. It also shows that

the improvement in recognition rate with the number of SLAMs is due to the rapid decrease in the rate of rejection.

Fig. 4.10 shows how the rate of substitution can be reduced at the expense of a rapid increase in rejection rate. (The arrows indicate the direction as the memory is filled). If the ratio of the cost, $K(j|i)$, of misclassifying a pattern to the cost, $K(i)$, of rejecting a pattern is known, an optimum cost-line can be drawn, e.g. on the figure is drawn the cost-line for

$$\frac{K(j|i)}{K(i)} = 2$$

and from the intersections, one can stop the training phase for a given size of PR to obtain the minimum loss.

The characteristics of a SLAM-PR can be summed up as follows:

(i) The rate of recognition increases rapidly, then tends to level off with increasing number of SLAMs (fig. 4.9a). This variation is reflected in the decrease in the rejection rate (fig. 4.9c).

(ii) There is a rapid decrease in the rate of substitution, accompanied

with an increase in rejection rate as the memory is filled (figs. 4.8b and 4.8c).

(iii) For any size of SLAM-PR the recognition rate increases to an optimum and then deteriorates with more training (fig. 4.8a).

(iv) For a given level of memory filled, the rate of substitution error is independent of the number of SLAMs (fig. 4.9b).

The above points except the last one will be discussed in section 5.1. The last point will be discussed in section 5.5.

4.6 Comparison with a Template-Matching Classifier

In the commercial field, devices using template matching techniques are already being used to recognize printed or typewritten characters. As a basis for comparison, the performance of a template-matching classifier using the same handwritten characters is estimated. The system, as shown in fig. 4.11, is

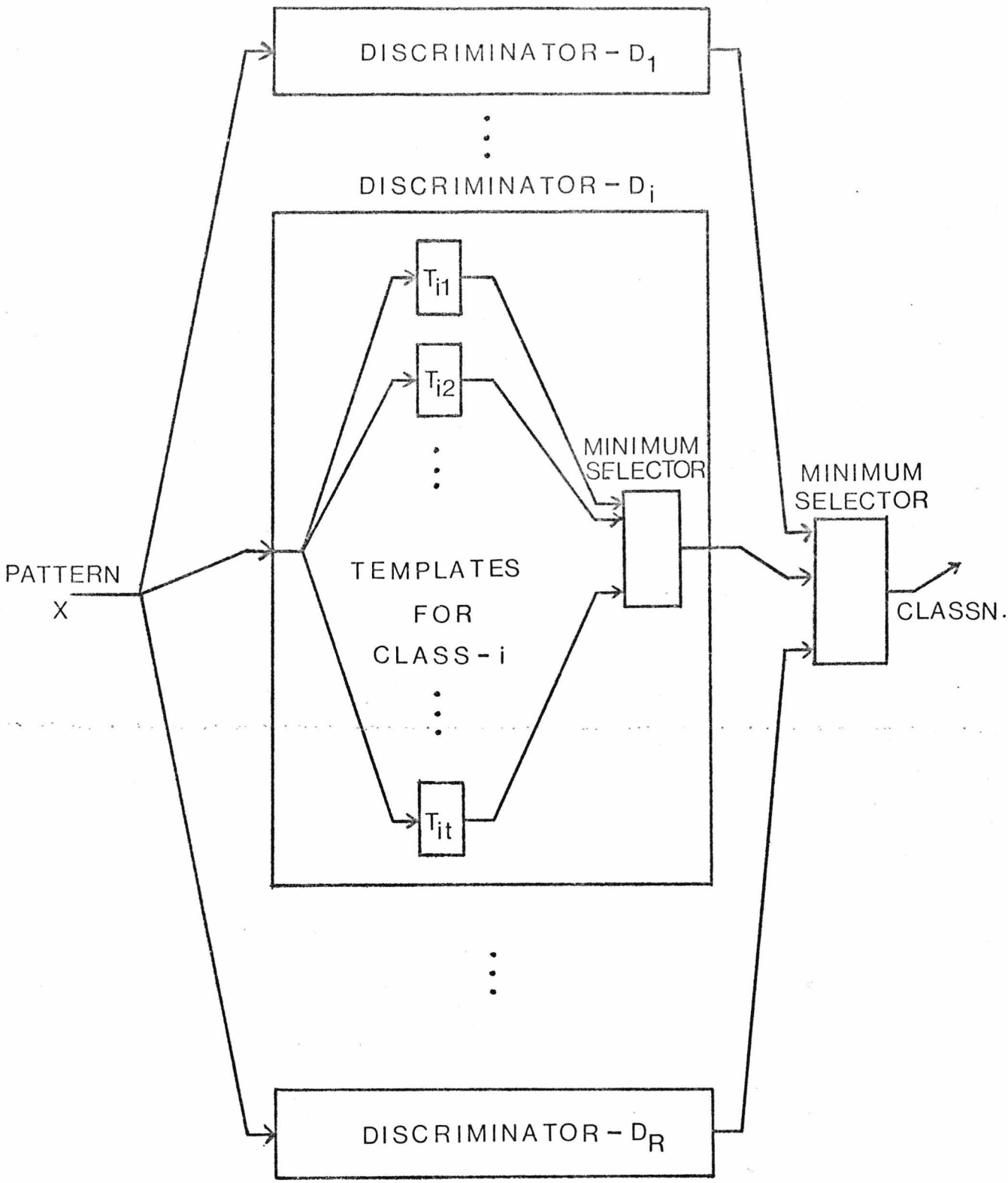

Fig. 4.11. Block Diagram of a Template-Matching Classifier.

simulated using various numbers of templates per discriminator. Assuming that the patterns in the training sets are all prototypes of their respective class, the templates are drawn from the training sets. The operations of the computer program for the simulation are shown in fig. 4.12.

The decision rule classifies an unlabelled pattern in the same class as the template which gives the maximum correlation. If two or more templates from different classes give an equal maximum correlation, the unknown pattern is rejected. A typical classification matrix for the template-matching classifier is shown in fig. 4.13.

The results for various numbers of templates per discriminator are shown in fig. 4.14. It shows the variation of recognition, substitution and rejection rates with number of templates.

The criterion for comparison between a SLAM-PR and a template-matching classifier is defined here as the rate of recognition with respect to the size of memory storage needed. Using SLAM-16 modules, which need 16 memory elements each, the number of memory storage bits required for a system of s SLAMs per discriminator is (sx10x16), for all ten discriminators.

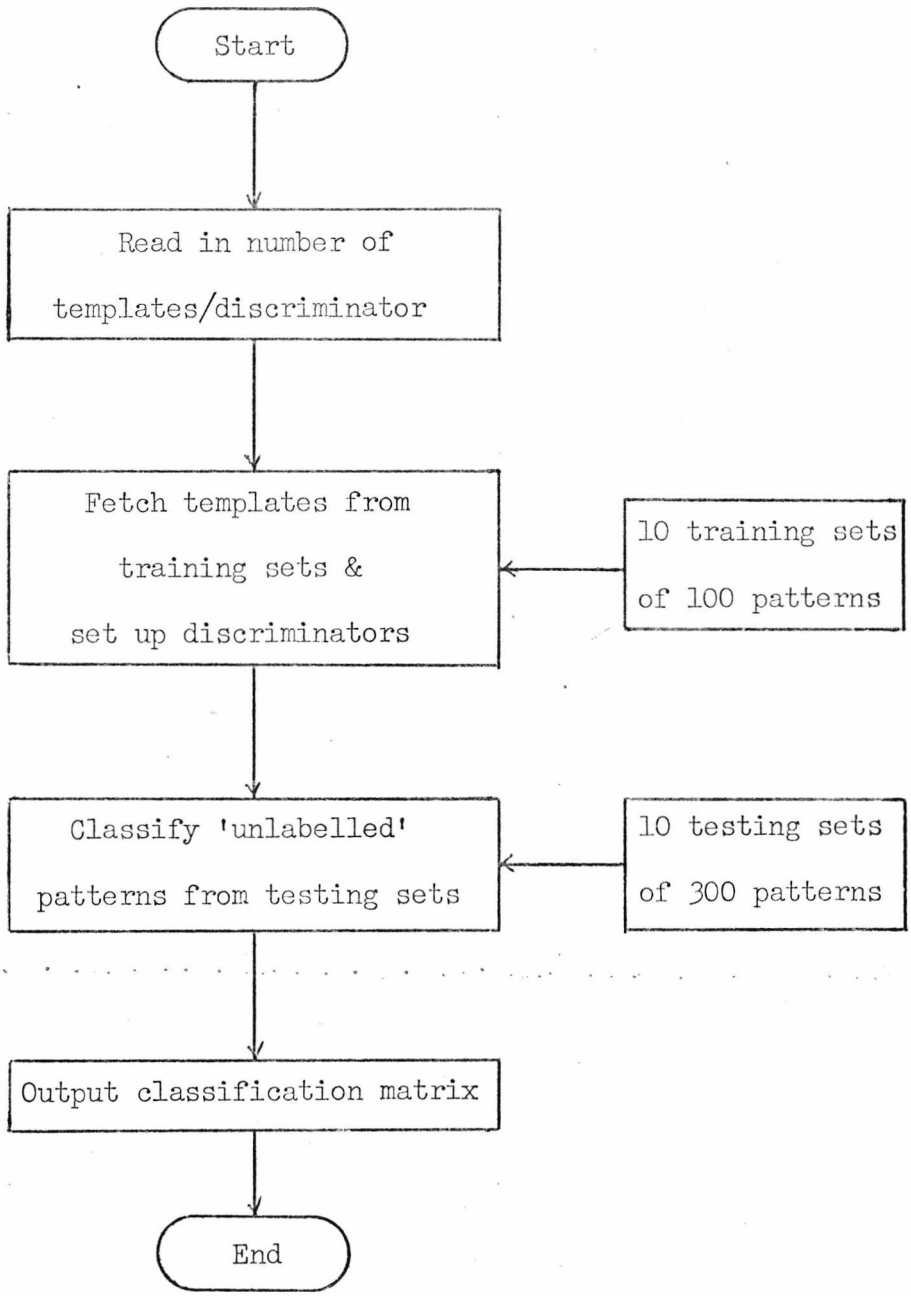

Fig. 4.12 Major Operations in Template-Matching Classifier Simulation.

ACTUAL PATTERN CLASS	CLASSIFICATION											
	1	2	3	4	5	6	7	8	9	0	REJ.	SUBS.
1	299	0	0	0	0	0	1	0	0	0	0	1
2	3	217	5	0	4	0	40	9	6	6	10	73
3	0	6	240	0	19	4	10	5	2	5	9	51
4	10	5	4	223	0	15	1	0	17	8	17	60
5	0	2	26	1	222	16	2	5	7	6	13	65
6	21	0	1	6	4	253	0	0	0	8	7	40
7	12	0	0	0	0	0	282	0	3	0	3	15
8	0	5	39	1	13	32	1	165	19	7	18	117
9	1	1	1	4	2	0	8	0	277	1	5	18
0	0	0	0	0	2	4	1	0	11	278	4	18

Overall Recognition Rate: 81.9 %

" Rejection " : 15.3%

" Substitution " : 2.9%

Fig. 4.13. A Typical Classification Matrix for the Template - Matching Classifier (20 Templates/Discriminator).

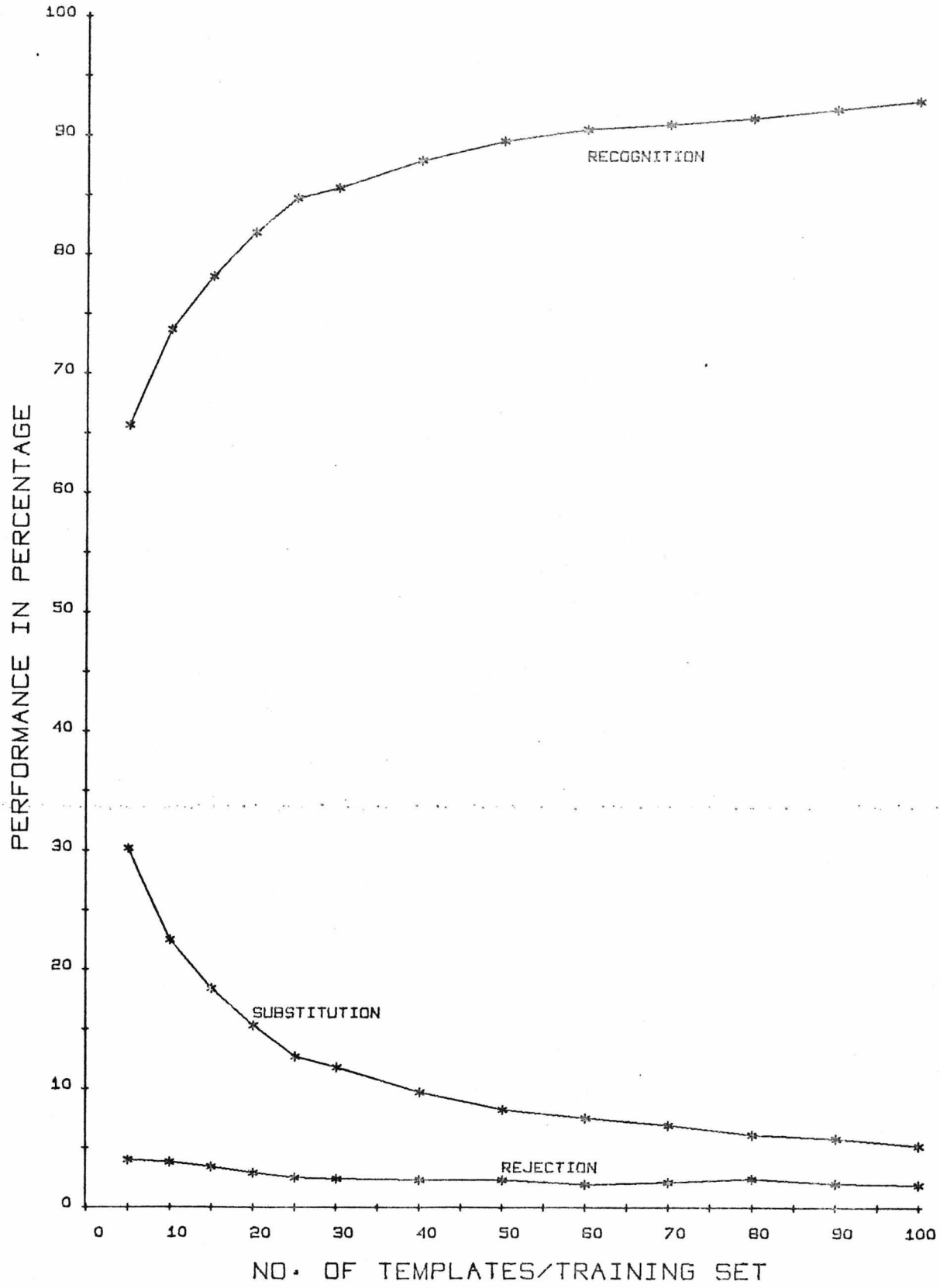

Fig. 4.14. Performance of Template-Matching Classifier.

For the template-matching classifier, each template is expressed by a 12x15 matrix and needs 180 memory storage bits. The number of memory storage bits for a classifier using t templates per pattern class is (tx10x180).

For a template-matching classifier, consisting of a given number of templates per pattern class, the performance is unique. However, for a given size of SLAM-PR, the classification is a function of the level of memory filled (fig. 4.8a). It is assumed that the latter can be trained to the optimum for the testing patterns. Fig. 4.15 shows the variation of the recognition rate for both systems with the size of memory storage.

One defines a simple loss function as one in which one unit is gained for a correct classification, one unit is lost for a misclassification and nothing is lost or gained for a rejection. Fig. 4.16 illustrates the performance for both systems when the simple loss function is assumed.

It can be seen that provided a SLAM-PR can be optimally trained, its performance is generally better than that of a template-matching classifier on the same handwritten data, for a given amount of memory

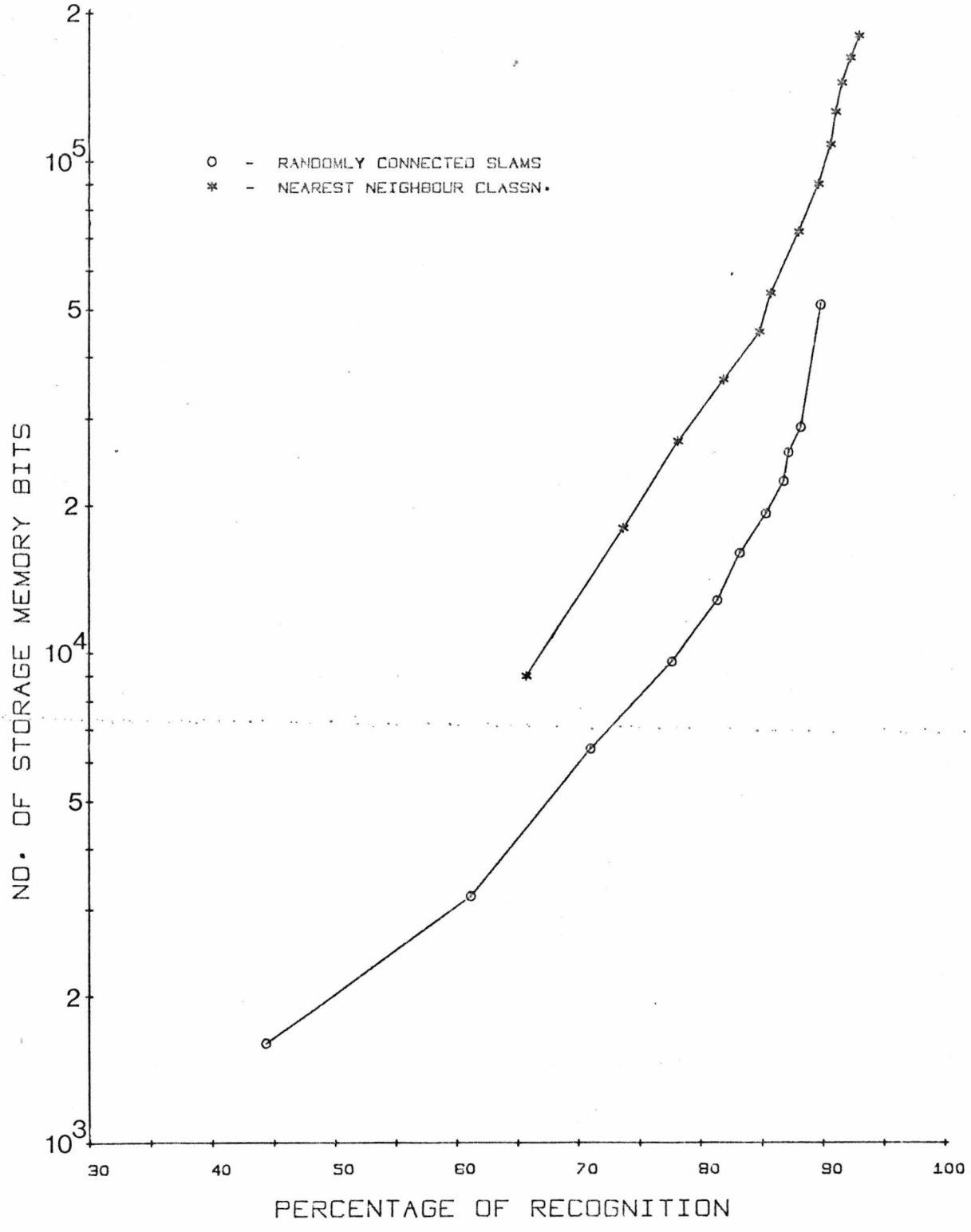

Fig. 4.15. Recognition Rate of Template-Matching Classifier & of Optimally Trained SLAM-PR against Storage.

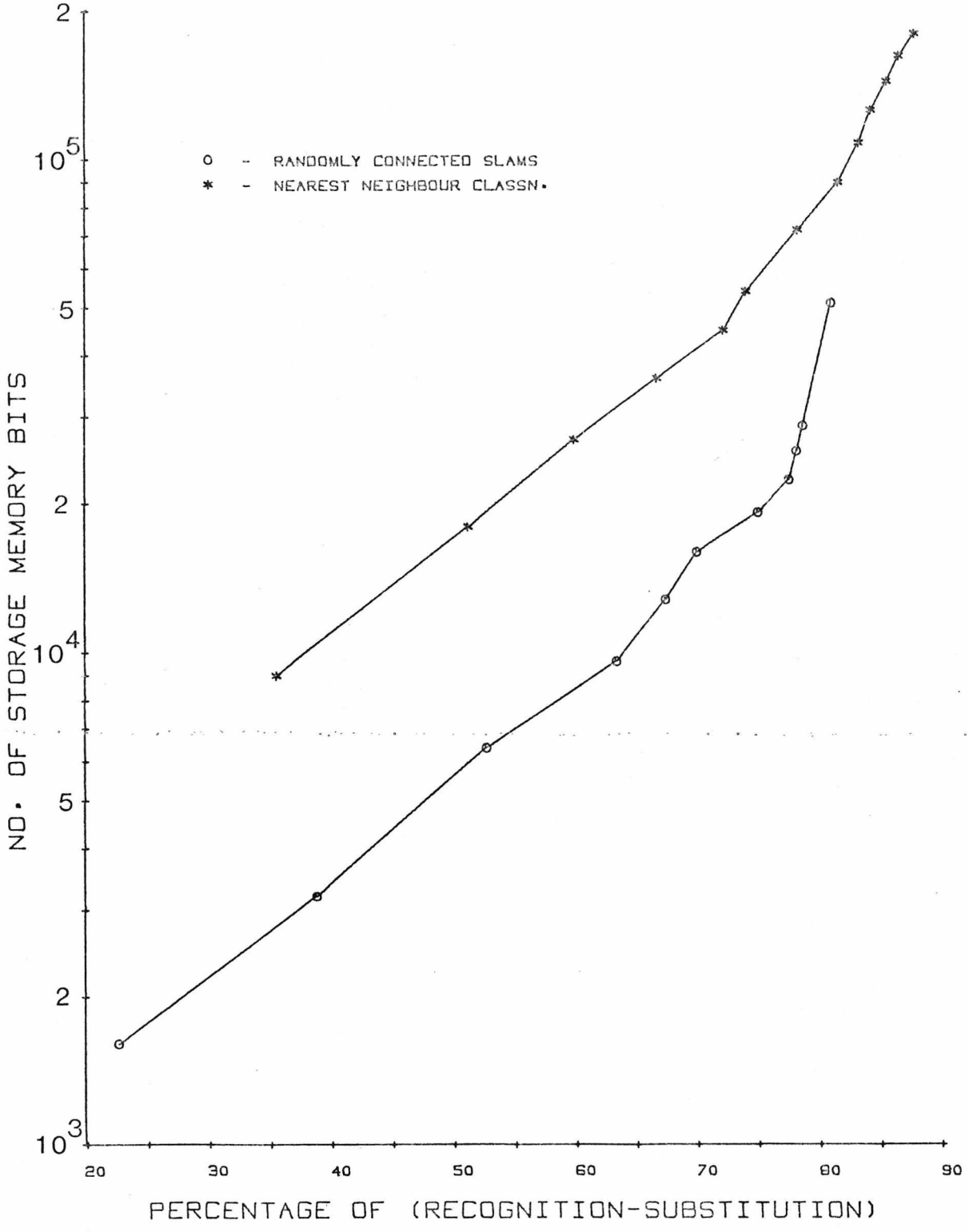

Fig. 4.16. Performance of Template-Matching Classifier & of Optimally Trained SLAM-PR, Assuming a Simple Loss Function.

storage. It must, however, be recalled that the size of SLAM modules for the comparison has been chosen arbitrarily. Similar experiments with PR using various sizes of SLAM modules have been reported by Ullmann (1969) where a constant training set size was assumed. This leads to a broader discussion which is tackled in the conclusion.

4.7 Pattern Recognizer Using Output-Weighted SLAMs

It is clear that for a certain level of memory filled, the SLAMs could be classified in terms of their usefulness in arriving at the final classification. This usefulness is related to the deviations from the average memory filled. This section reports on various arbitrary "usefulness" assignments in terms of a weighting of the SLAM outputs as functions of memory filled.

To this end, a PR similar to the basic SLAM-PR (section 4.5), with a slight modification in the SLAM structure is considered, whereby the output of each SLAM is weighted as a function of the number of bits set in the particular SLAM. A PR, using such modified SLAMs, is referred to as an output-weighted SLAM-PR, and is shown in fig. 4.17. Various

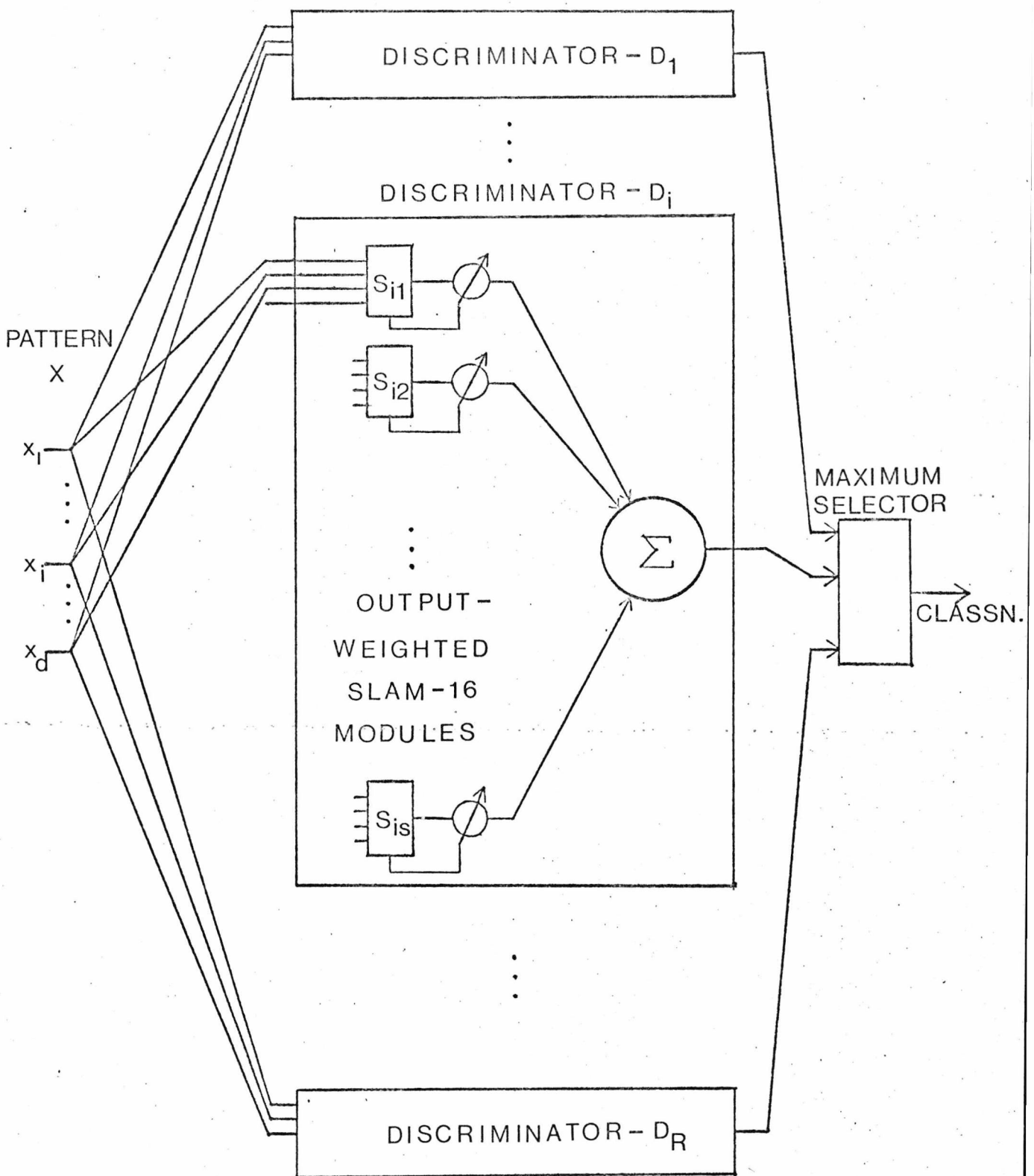

Fig. 4.17. Block Diagram of an Output-Weighted SLAM-PR.

weighting schemes, illustrated in fig. 4.18, are implemented and are as follows:

(1) 'Uniform' weighting (same as in basic SLAM-PR)

$$w(i) = 1 \quad \text{for } i = 1,\ldots,16$$

w(i) is the weighting value for the output when the SLAM has i memory bits set.

(2) 'Ramp' weighting

$$w(i) = i \quad \text{for } i = 1,\ldots,16$$

(3) 'Antiramp' weighting

$$w(i) = 17-i \quad \text{for } i = 1,\ldots,16$$

(4) 'Entropy' weighting

$$j = \frac{i}{16} \quad \text{for } i = 1,\ldots,16$$

$$w(i) = \{j \log_2(\tfrac{1}{j}) + (1-j)\log_2(\tfrac{1}{1-j})\} \times 100$$

(5) 'Peak' weighting

$$j = i-1 \quad \text{for } i = 1,\ldots,8$$

$$j = 16-i \quad \text{for } i = 9,\ldots,16$$

$$w(i) = 2^j$$

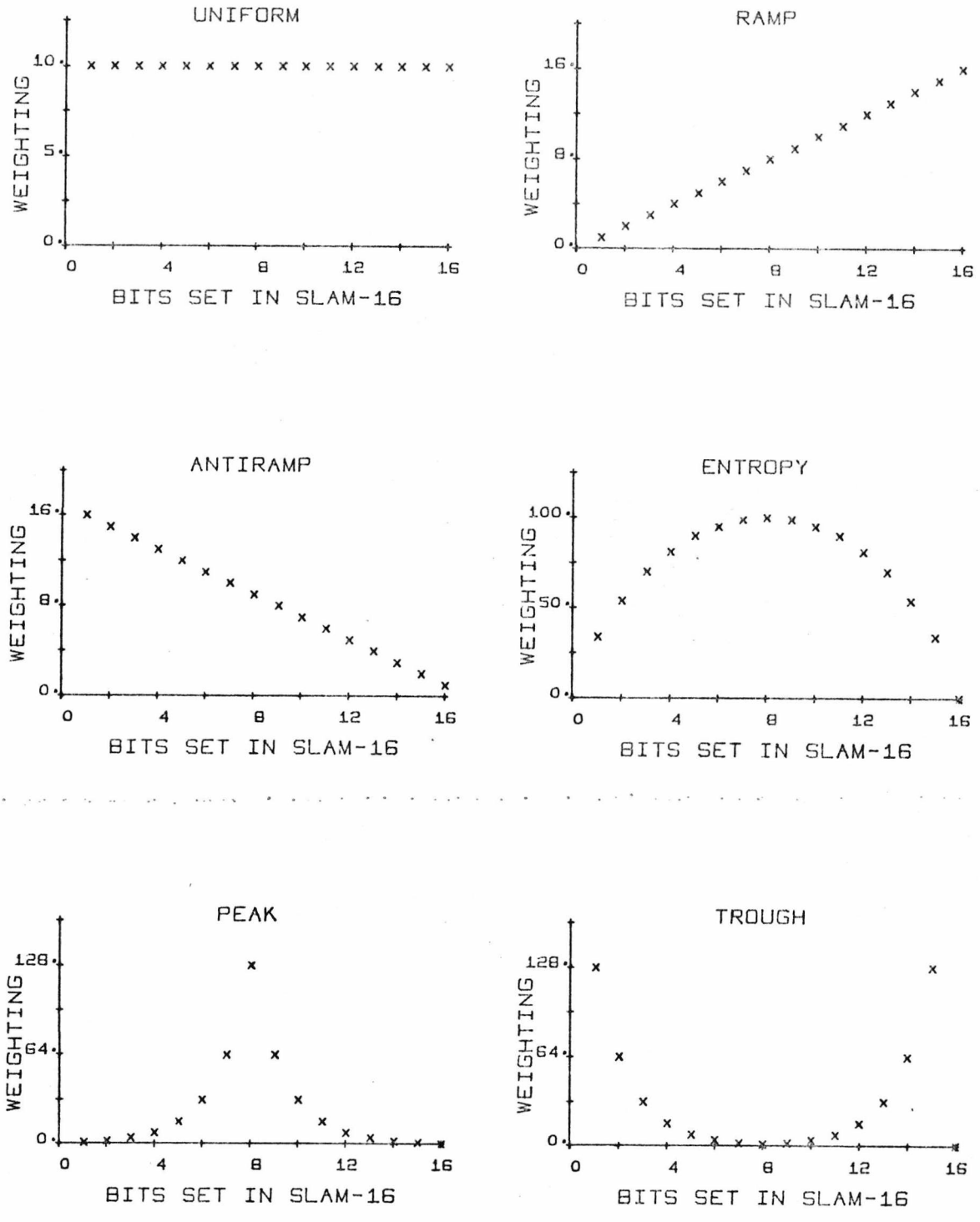

Fig. 4.18. WEIGHTING SCHEMES FOR 'OUTPUT WEIGHTED' SLAMS

(6) 'Trough' weighting

$$j = |8-j| \quad \text{for } i = 1,\ldots,15$$

$$w(i) = 2^j$$

$$w(16) = 0$$

A PR, which implements the 'uniform' weighting scheme, is equivalent to the basic SLAM-PR and serves as a basis for comparison for the other output-weighted SLAM-PRs. The weighting schemes are arbitrarily composed, e.g. the weighting value w(16) for the 'trough' weighting scheme is set to zero to preserve the symmetry of the distribution of the weighting values with number of bits set (fig. 4.18). A PR implementing say the 'peak' scheme will be referred to as a Peak-PR, and similarly for the other output-weighted SLAM-PRs.

In the experimentation, the patterns from the training sets are also tested. The classification of the training patterns is independent of the weighting schemes since a discriminator which has been trained on a pattern will always give maximum response when tested on that particular pattern. The resulting outcome for a training pattern can only be either a correct classification or a rejection. A rejection occurs when a discriminator, other than the one which has been trained on the pattern, also gives

maximum response. If the training patterns are representative of all the possible patterns, then the recognition rate for the training patterns is the upper bound to the performance of the various systems.

Simulations of the different output-weighted SLAM-PRs and for different sizes (10, 20,..., 160 output-weighted SLAMs/discriminator) are carried out. The results are plotted:

Figs. 4.19a,b,c,d,e show the variation in the rate of recognition for various sizes of PR at different levels of memory filled.

Figs. 4.20a,b,c,d,e show the variation in the rate of misclassification for various sizes of PR at different levels of memory filled.

Figs. 4.21a,b,c,d,e show the variation in the rate of rejection for various sizes of PR at different levels of memory filled.

From the results, the characteristics of an output-weighted SLAM-PR are basically similar to those of a basic SLAM-PR (section 4.5), there are, however, two additional points which are:

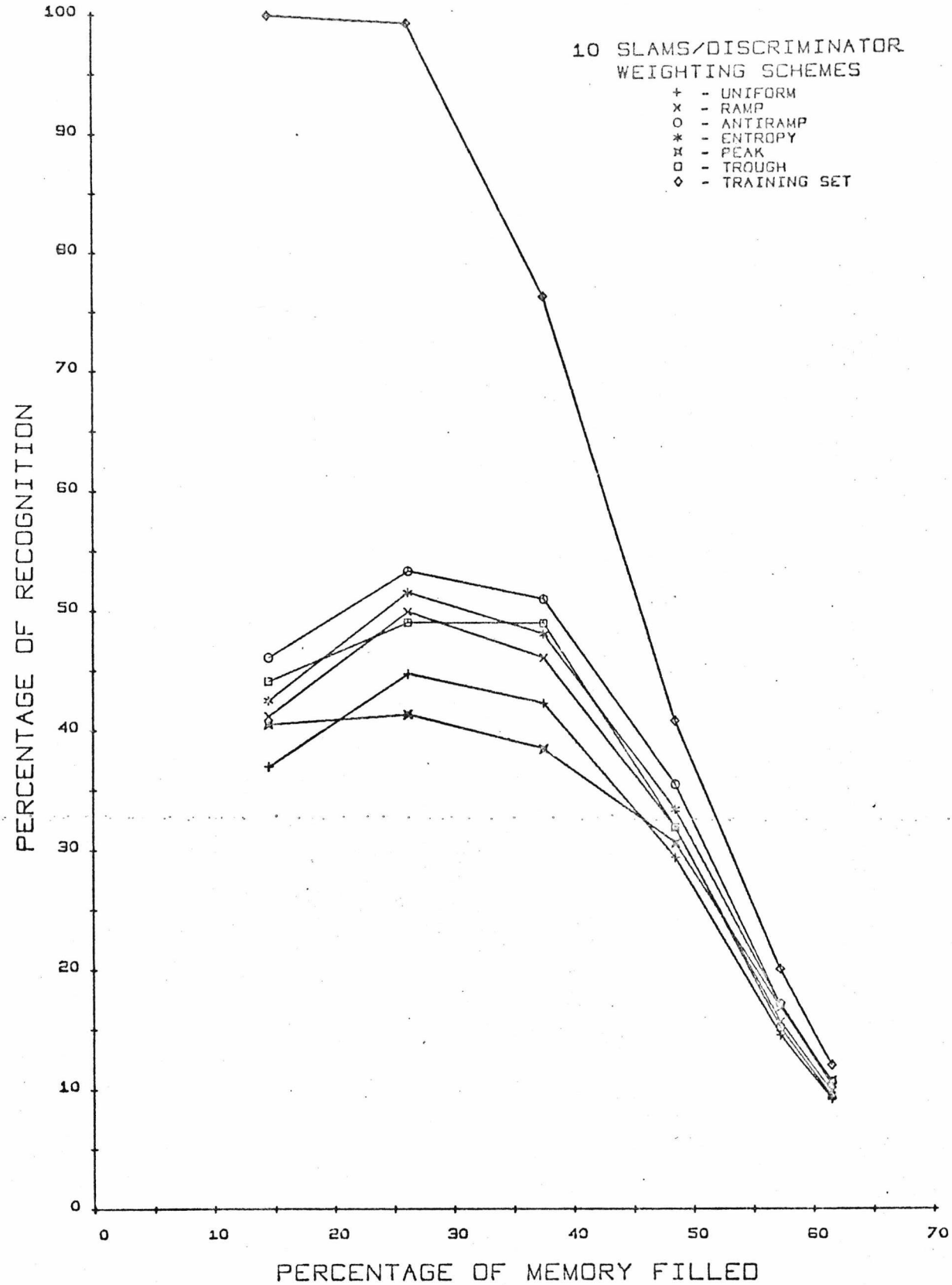

Fig. 4.19a. Recognition Rate versus Memory Filled (10 SLAMs).

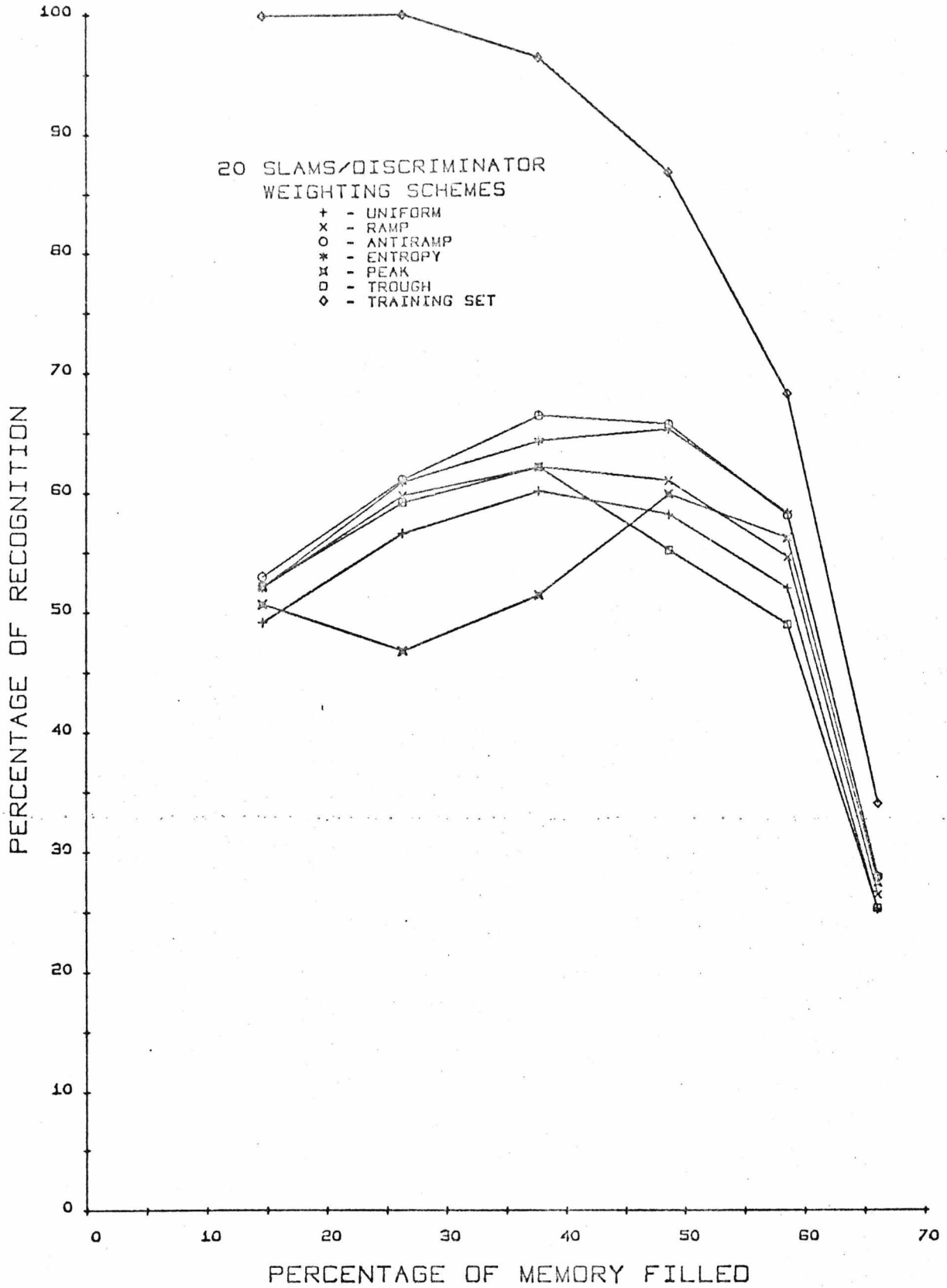

Fig. 4.19b. Recognition Rate versus Memory Filled (20 SLAMs).

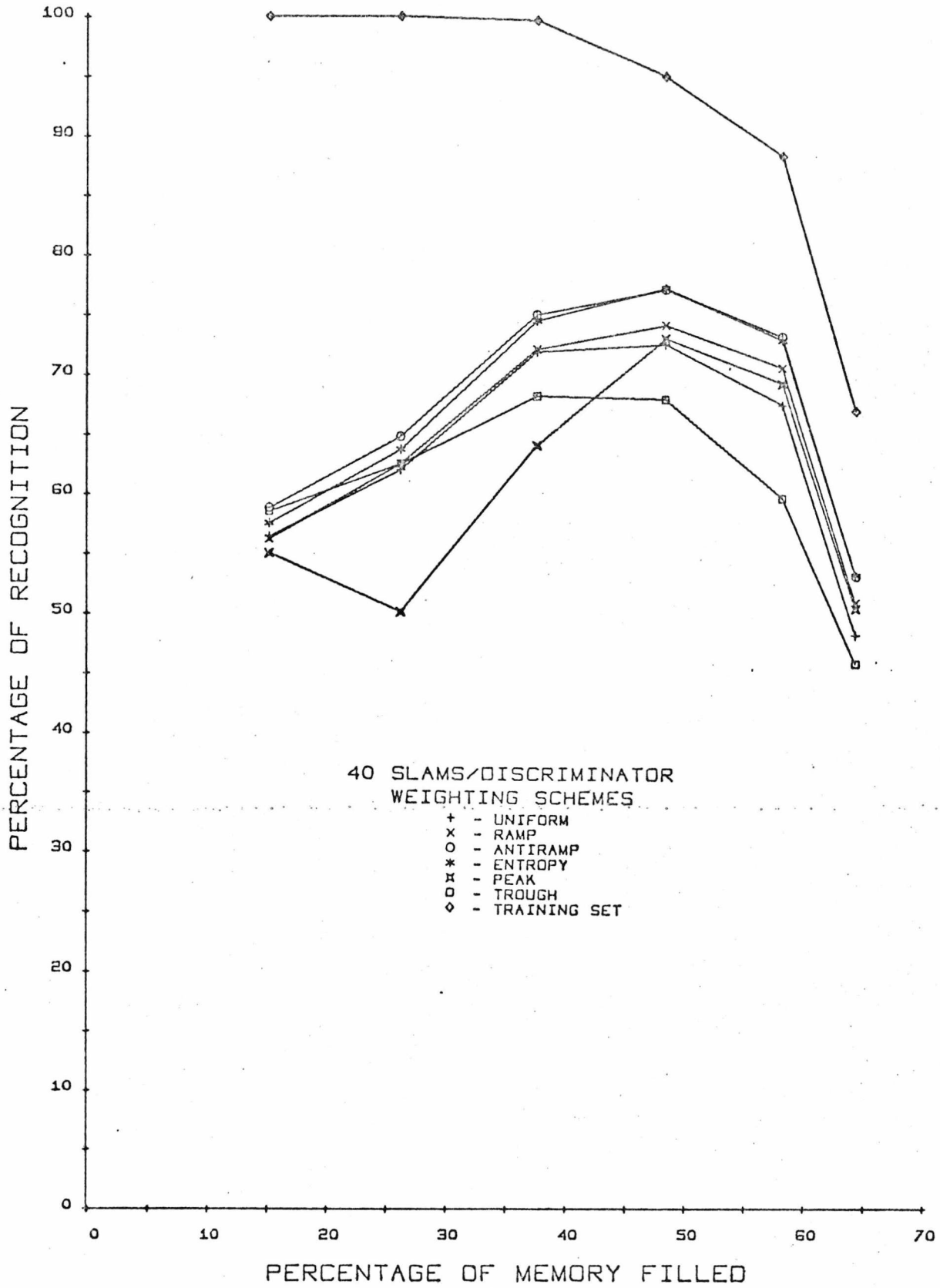

Fig. 4.19c. Recognition Rate versus Memory Filled (40 SLAMs).

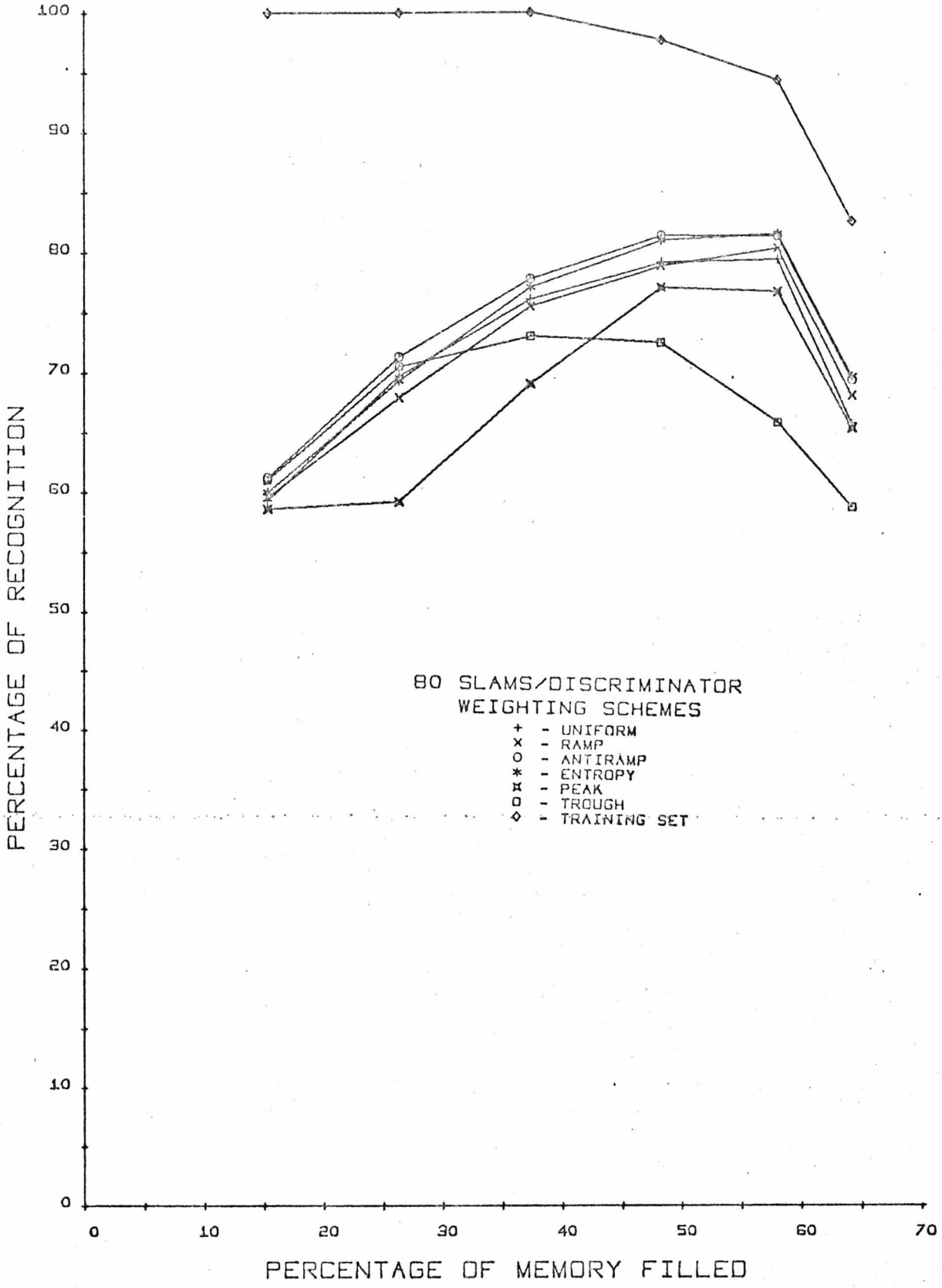

Fig. 4.19d. Recognition Rate versus Memory Filled (80 SLAMs).

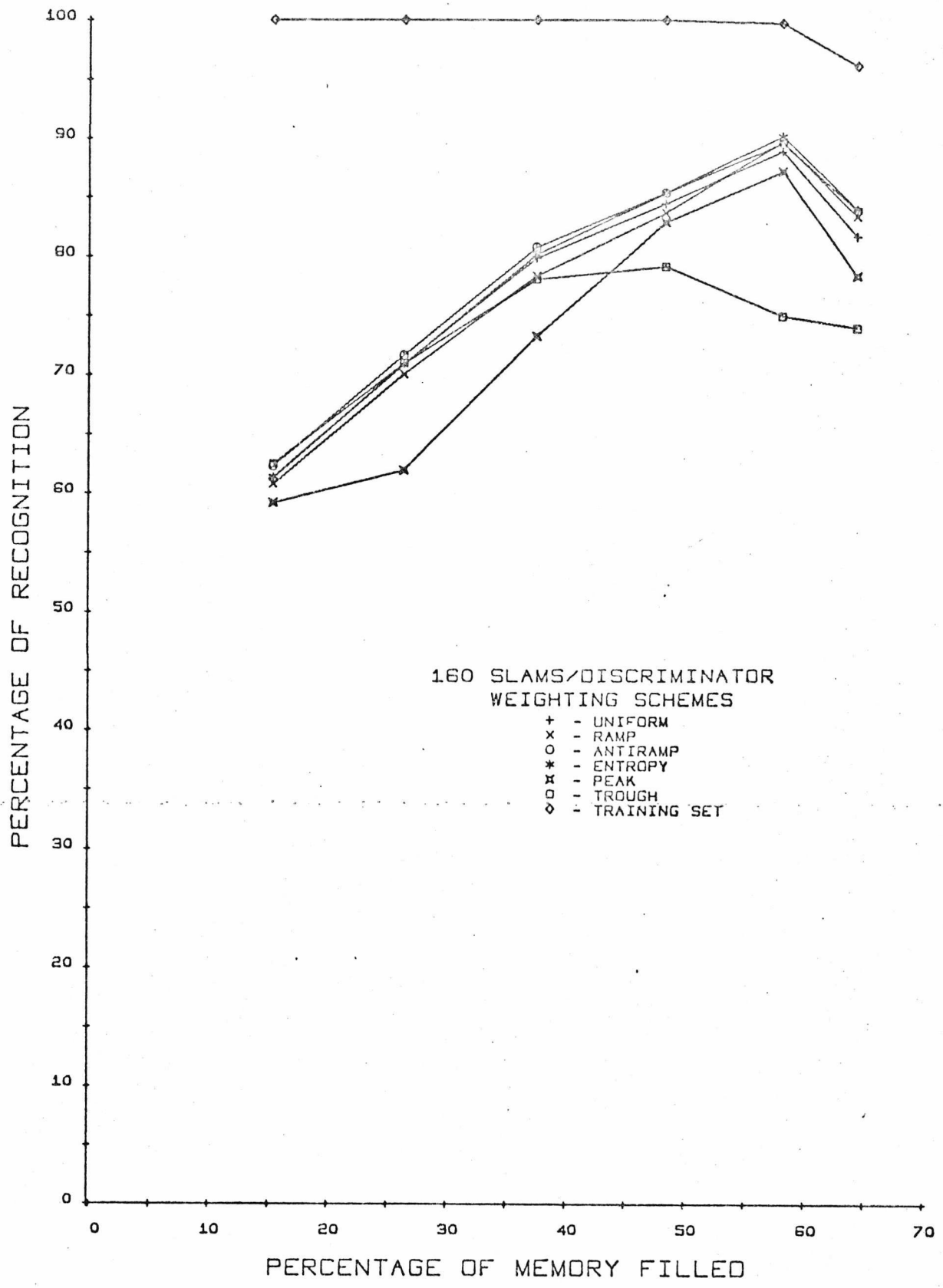

Fig. 4.19e. Recognition Rate versus Memory Filled (160 SLAMs).

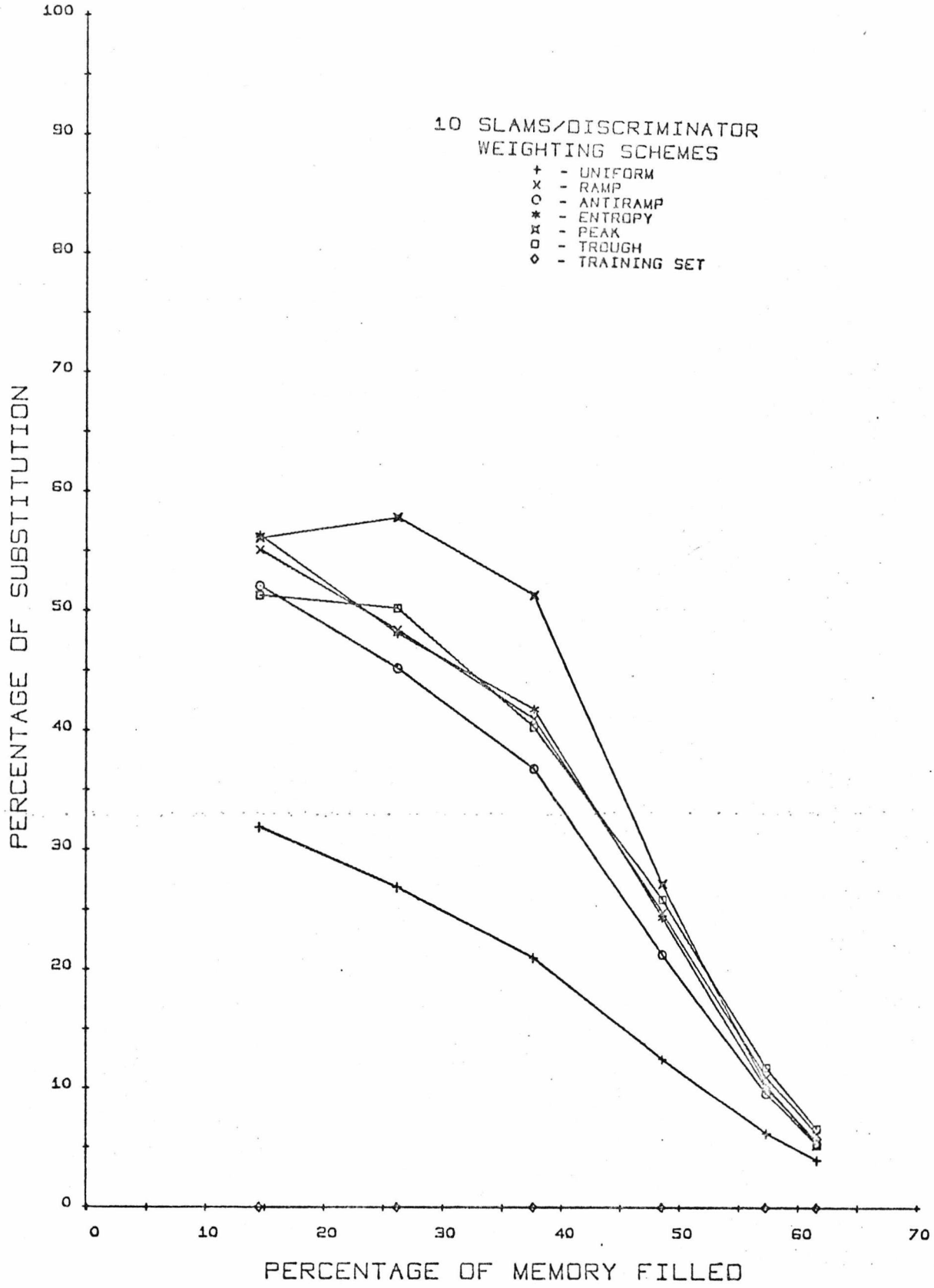

Fig. 4.20a. Substitution Rate versus Memory Filled (10 SLAMs).

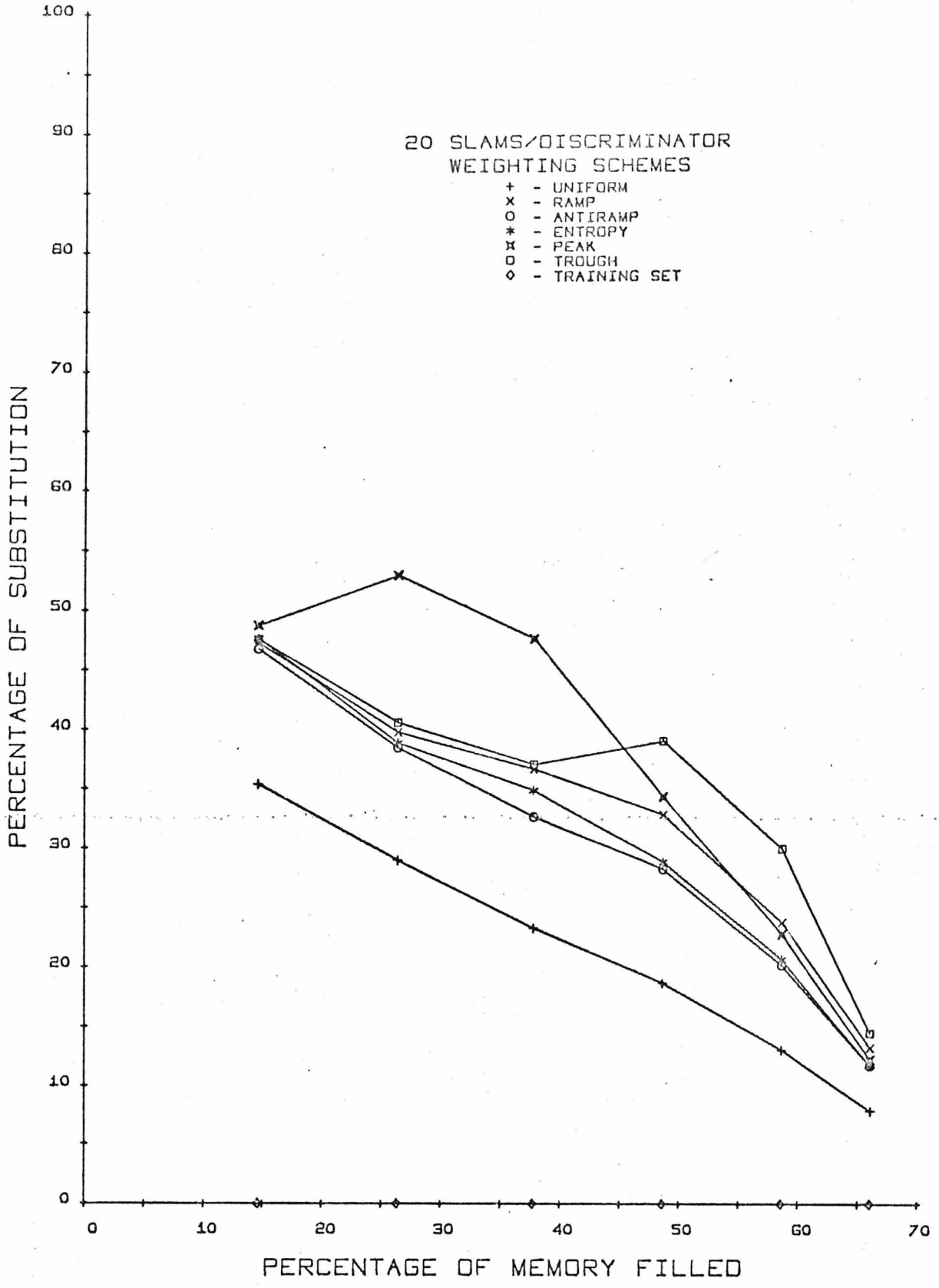

Fig. 4.20b. Substitution Rate versus Memory Filled (20 SLAMs).

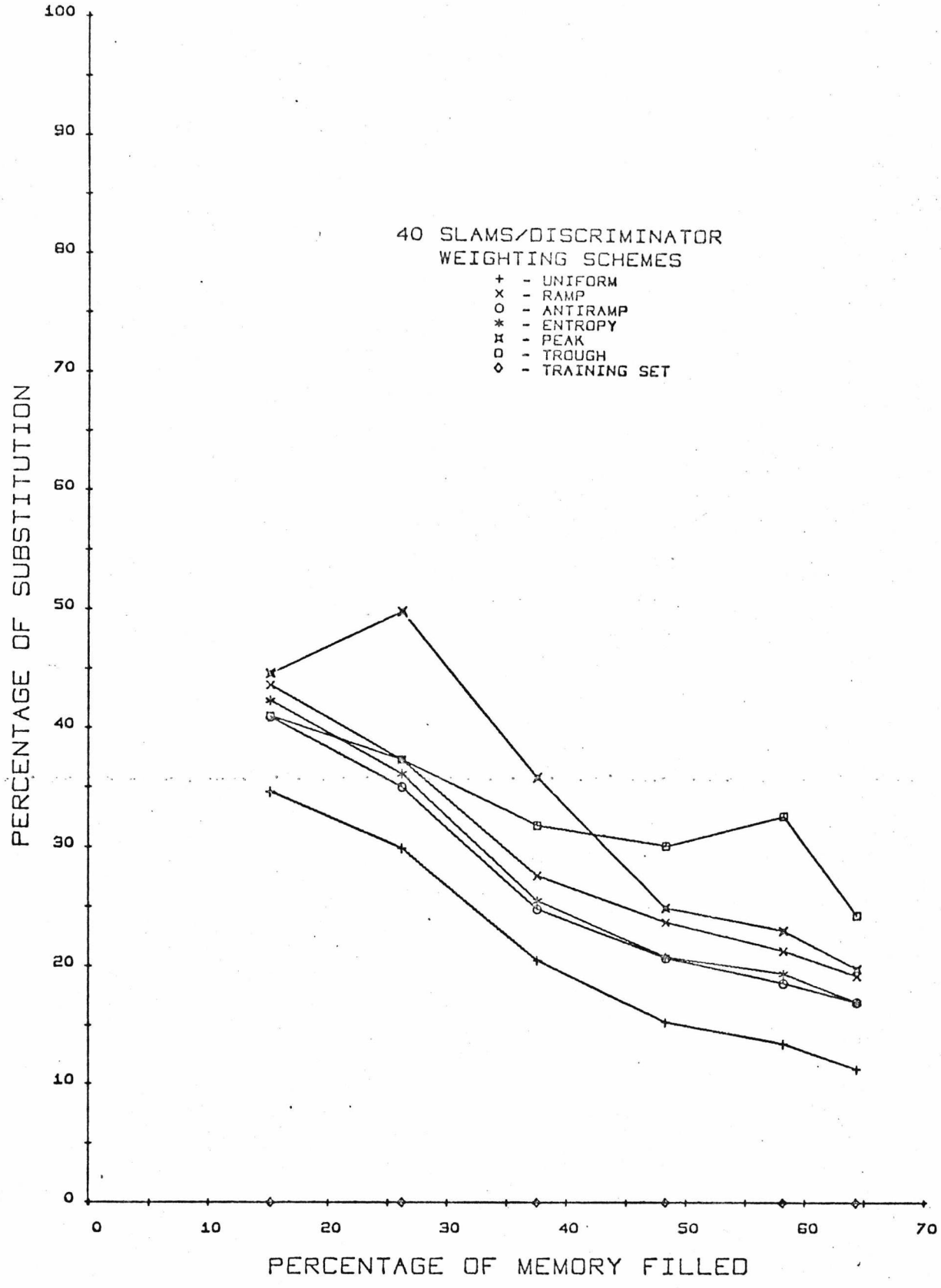

Fig. 4.20c. Substitution Rate versus Memory Filled (40 SLAMs).

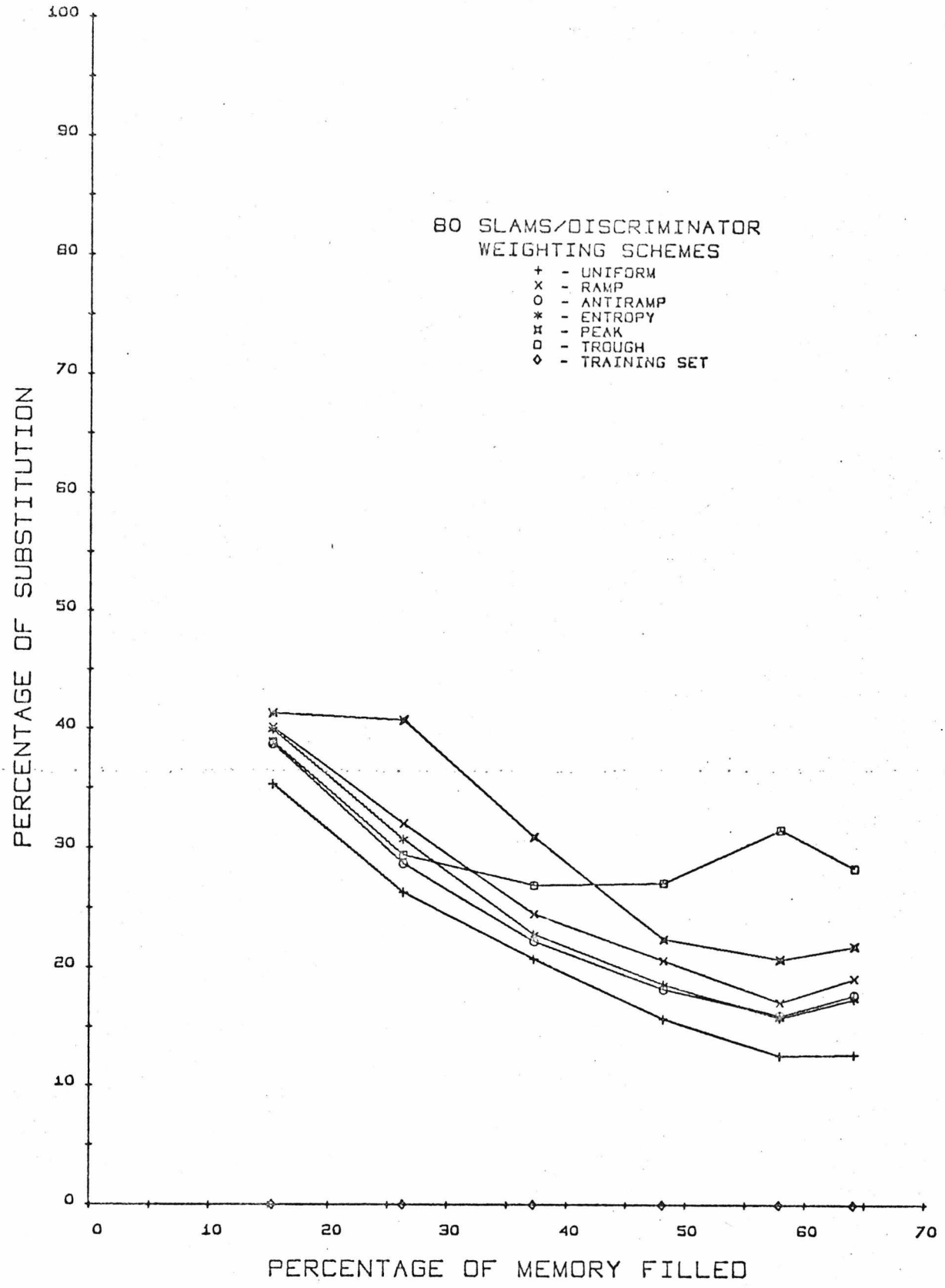

Fig. 4.20d. Substitution Rate versus Memory Filled (80 SLAMs).

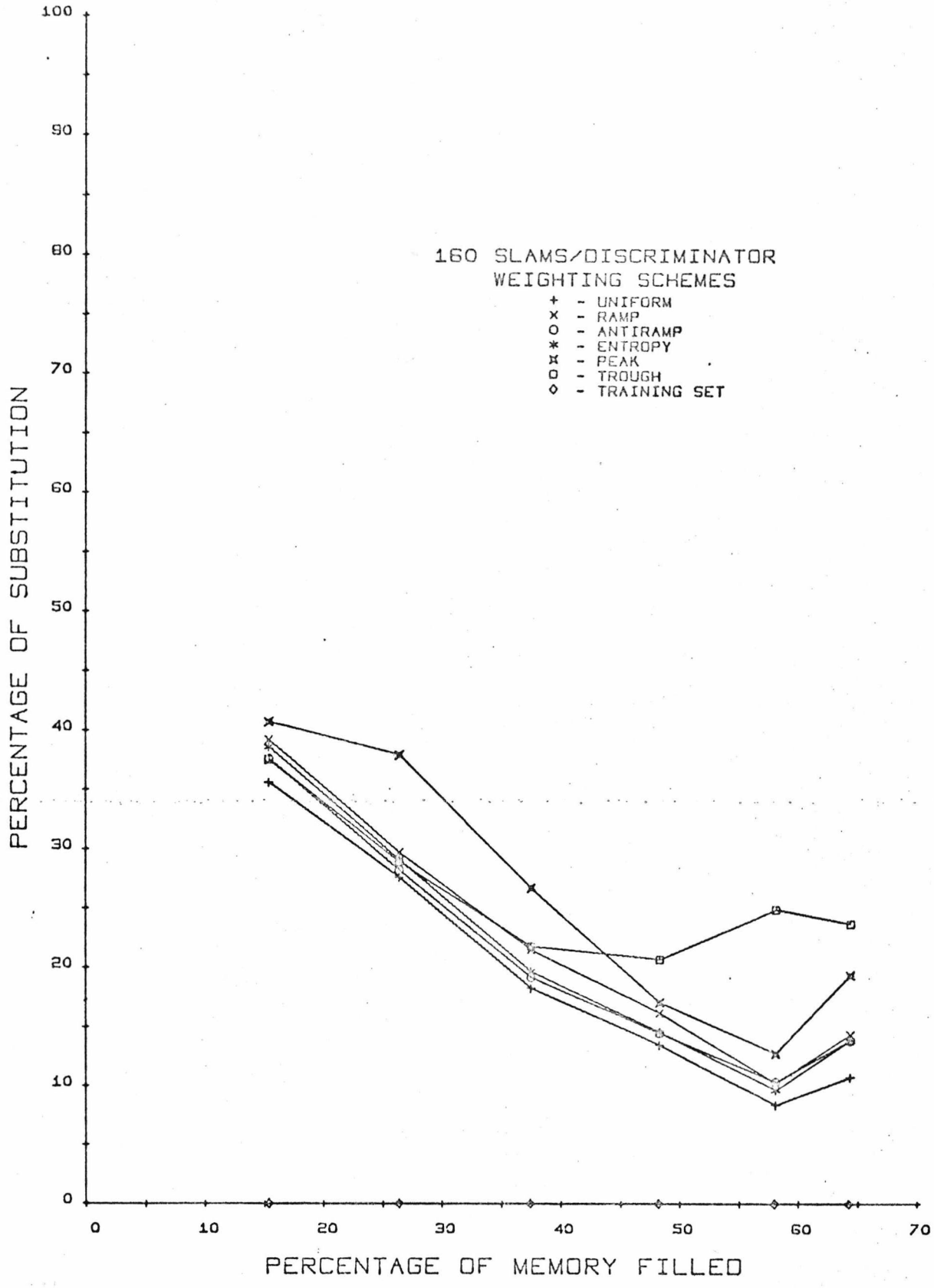

Fig. 4.20e. Substitution Rate versus Memory Filled (160 SLAMs).

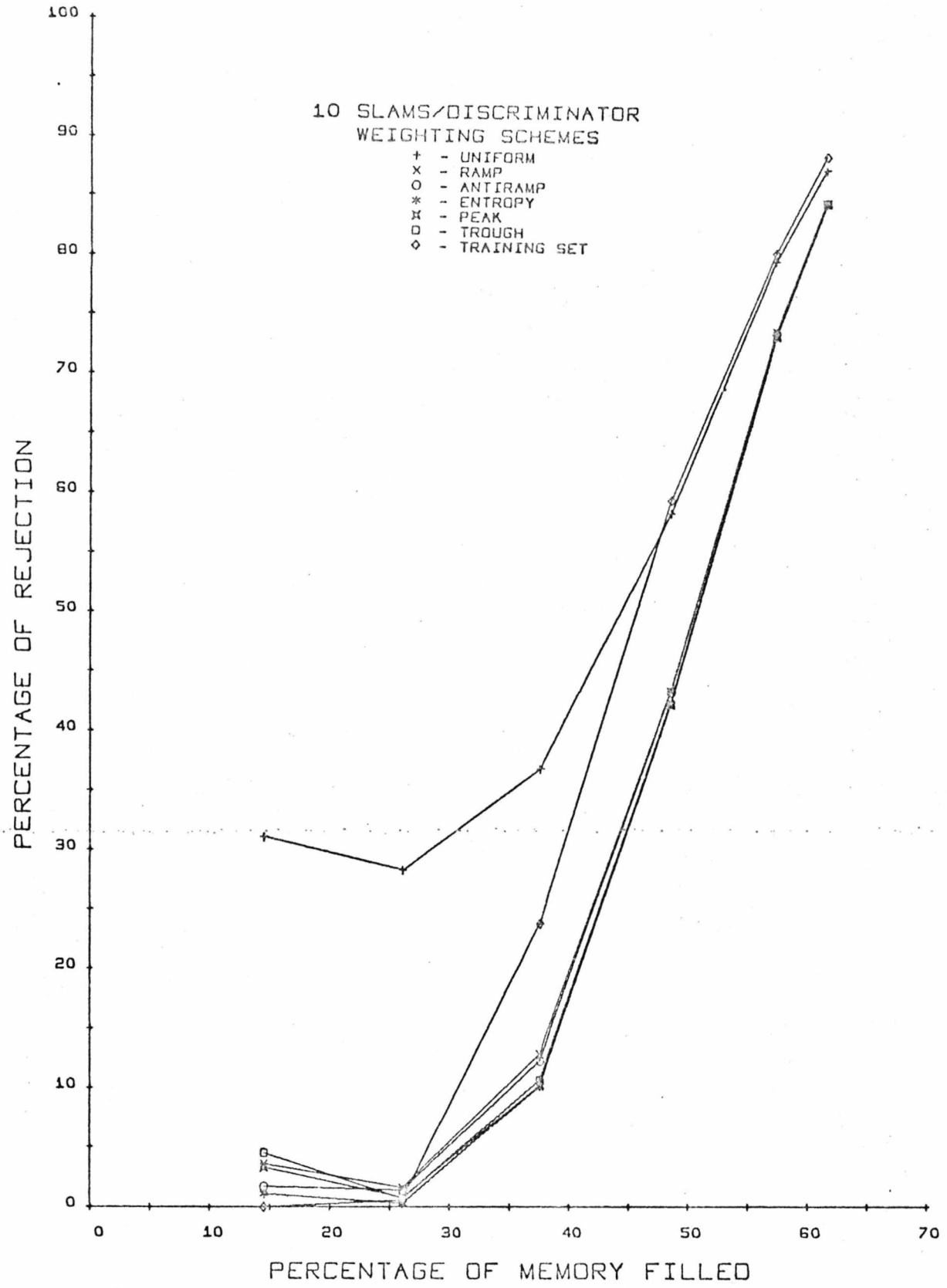

Fig. 4.21a. Rejection Rate versus Memory Filled (10 SLAMs).

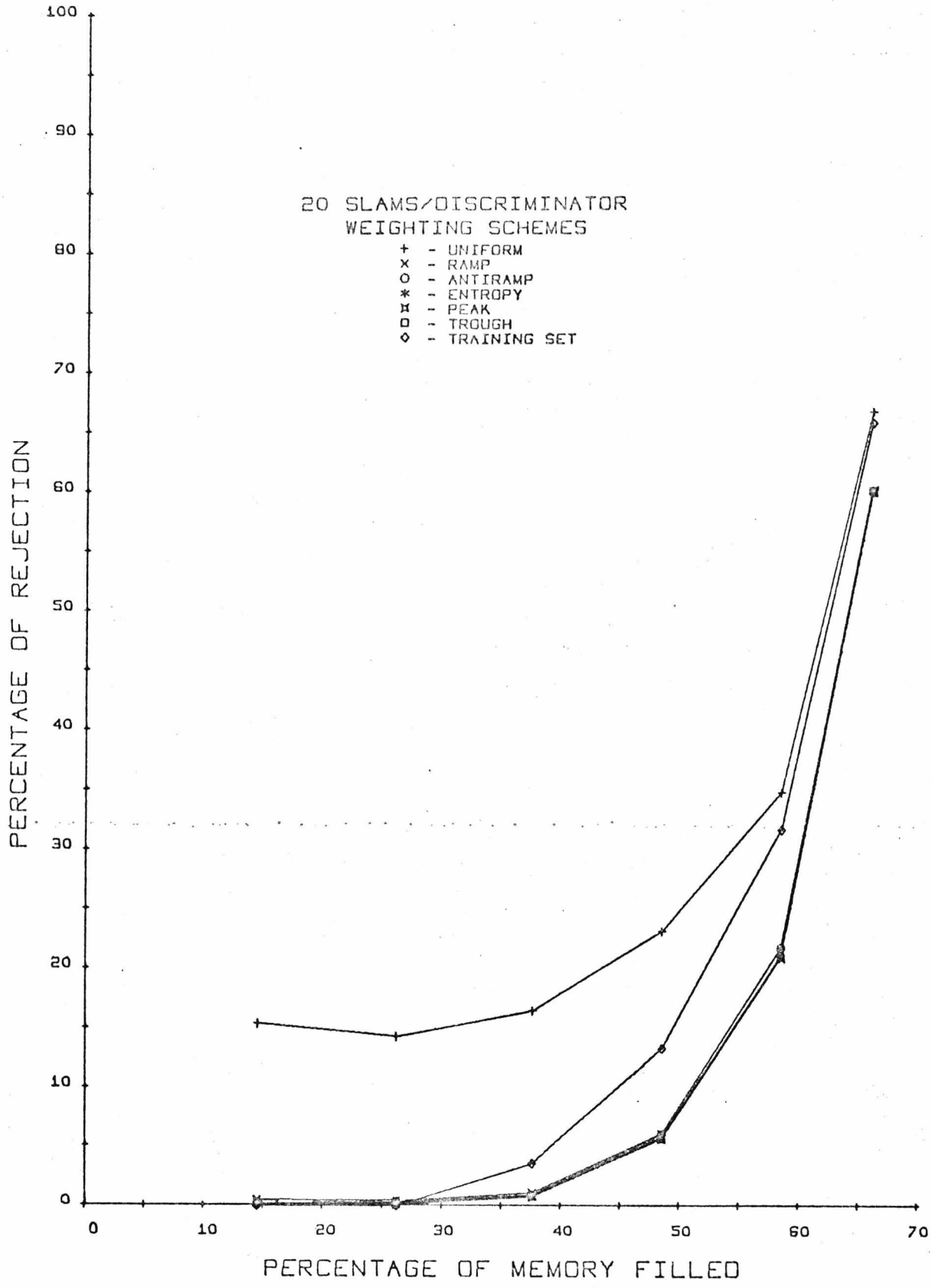

Fig. 4.21b. Rejection Rate versus Memory Filled (20 SLAMs).

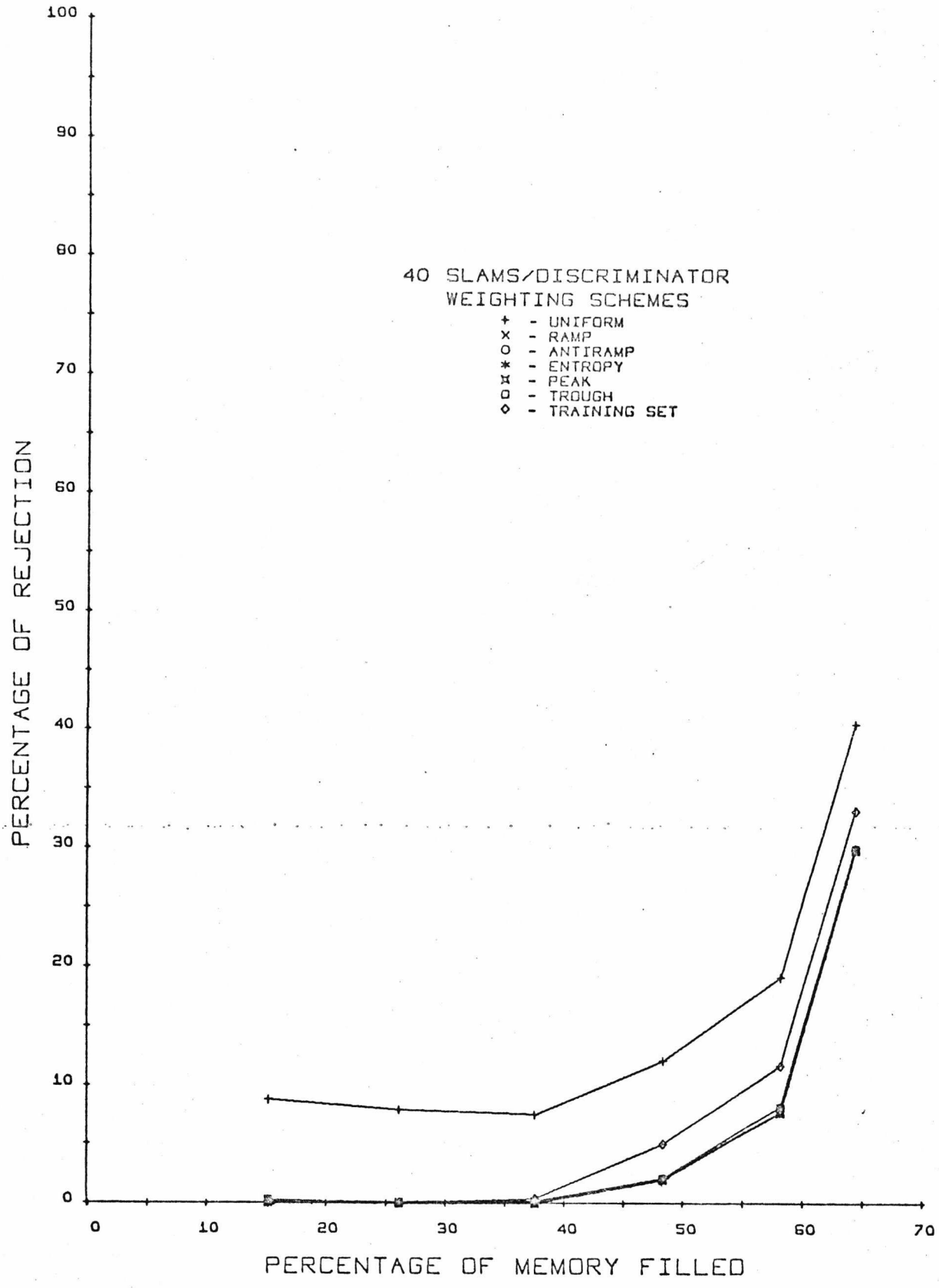

Fig. 4.21c. Rejection Rate versus Memory Filled (40 SLAMs).

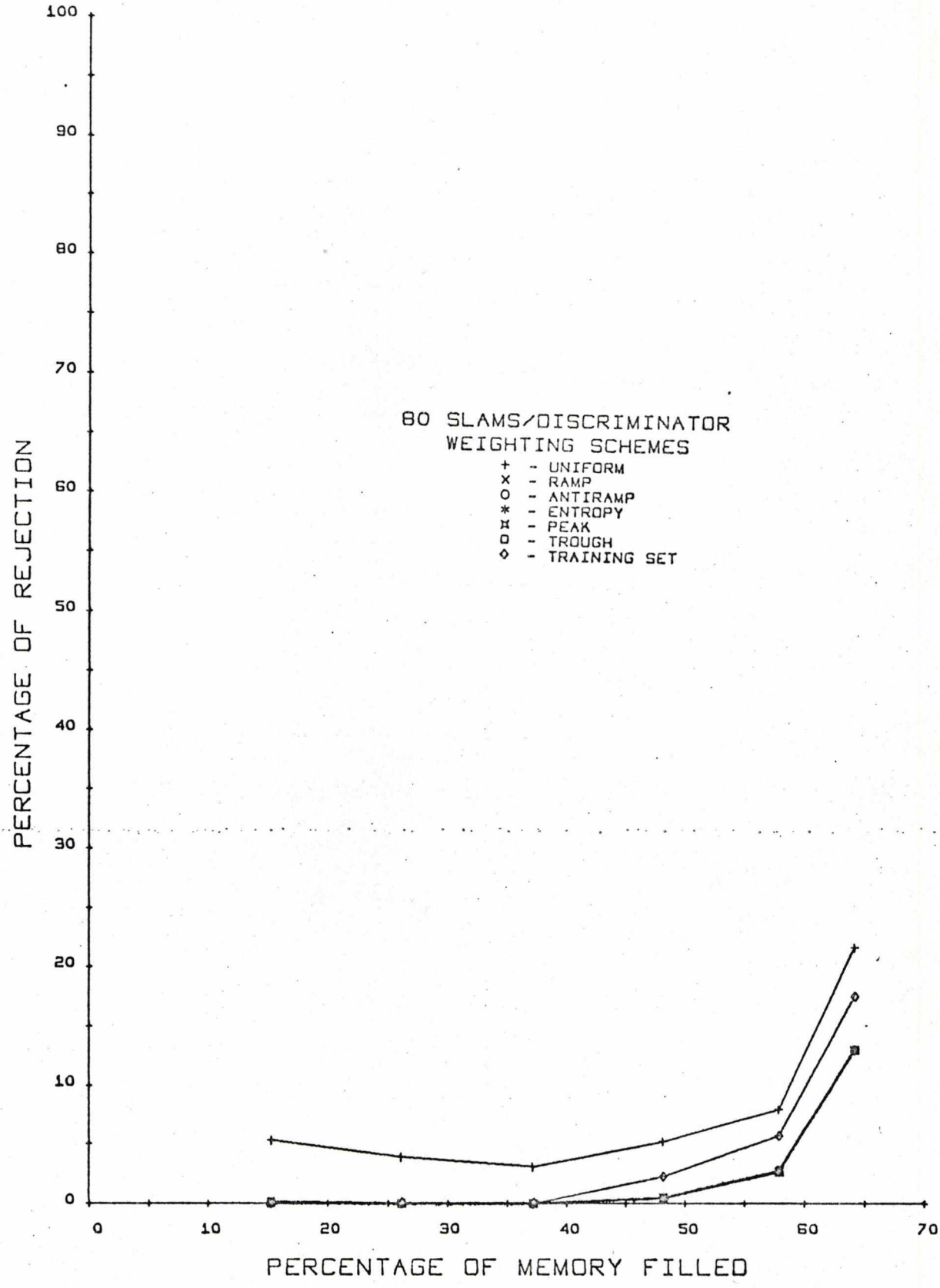

Fig. 4.21d. Rejection Rate versus Memory Filled (80 SLAMs).

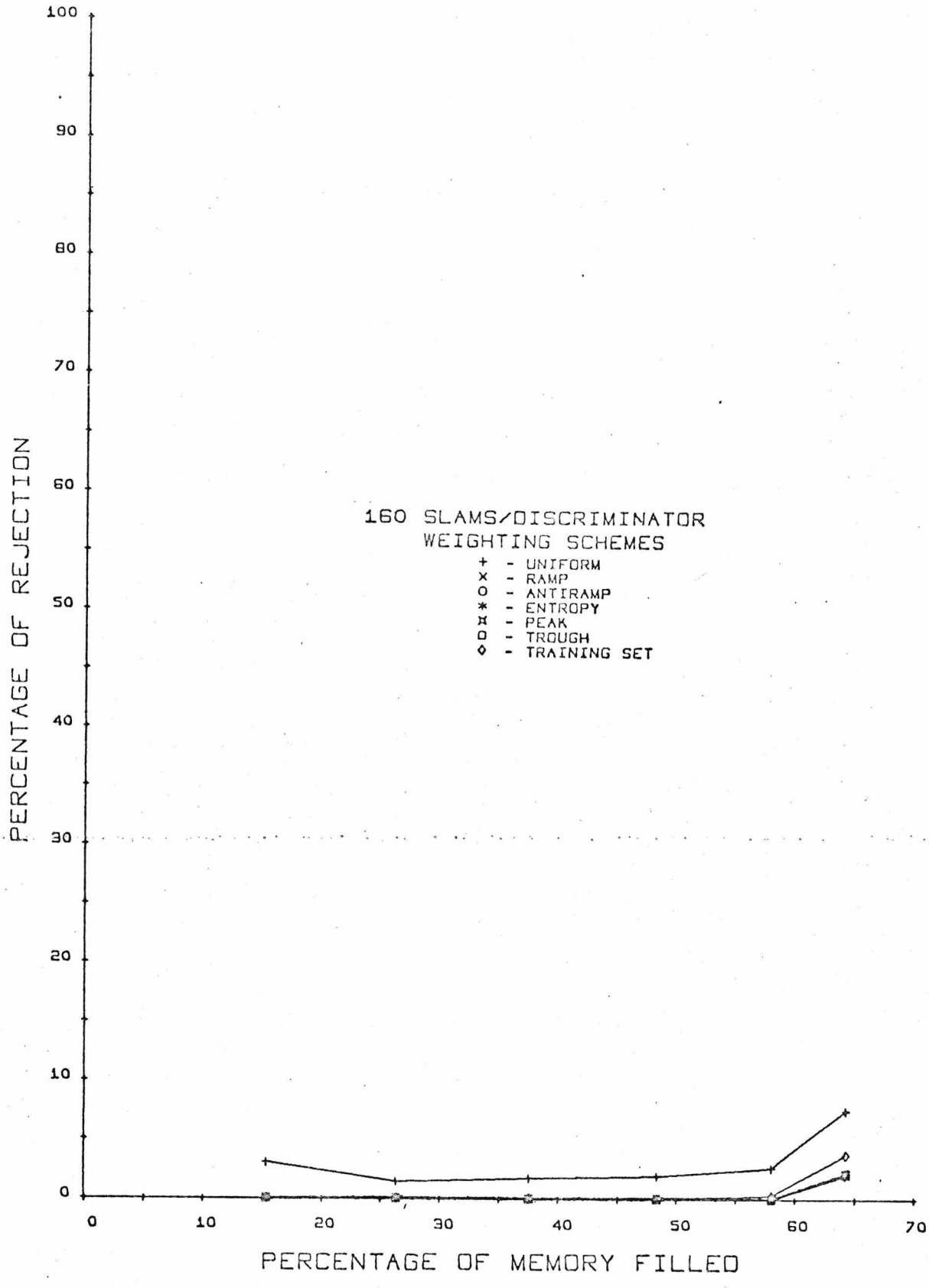

Fig. 4.21e. Rejection Rate versus Memory Filled (160 SLAMs).

(i) The Uniform-PR consistently generates more rejections than the remaining PRs.

(ii) Variations in the performance for the different output-weighted SLAM-PRs are entirely due to the difference in the weighting schemes, as the same set of random connections are used. The noticeable variations are:

(1) The recognition rate curves for the Peak-PR and Trough-PR consistently cross each other at about half memory filled.

(2) The recognition rate for the Antiramp-PR is generally better than the remaining systems.

The above two points will be discussed in sections 5.1 and 5.5 respectively.

4.8 Clustered SLAM Pattern Recognizer

It has been suggested by several workers that a completely random choice of features prevents the SLAMs from becoming sensitive to local events such as edges, filled in or empty spaces, etc. .... To this

end, a system using 'localized' or 'clustered' SLAMs is simulated.

A clustered SLAM is one whose inputs are connected within a 3x3 submatrix; one input being connected to the centre of the submatrix and the remaining inputs are connected randomly to the remaining retinal points in the submatrix. Hence, a SLAM-16 would have 3 inputs distributed randomly in the remaining 8 retinal points. Another constraint which is imposed, is that no two inputs to the same SLAM are connected to the same retinal point. For a 15x12 retina matrix, 180 clustered SLAMs are used for each discriminator, one for each retinal point. For a clustered SLAM which is connected to a corner of the retina, there is only one possible 4-tuple.

The performance of a PR using clustered SLAMs is compared to that of a basic SLAM-PR of the same size (180 randomly connected SLAMs per discriminator). The results are shown in fig. 4.22. A consequence of clustering the n-tuples is that the optimum occurs at a lower level of memory filled and the memory is more difficult to fill completely.

For clustered SLAMs connected to areas of low retinal activity, e.g. in the corners of the retina, the probability of encountering more than one state

of the n-tuple is low; while the n-tuples in the high activity regions are likely to have more different states than a random non-clustered n-tuple. This is shown in fig. 4.23 which illustrates a typical distribution of the SLAMs with the number of bits set after training, for both a PR with clustered SLAMs and a PR with non-clustered SLAMs. The distributions for the individual discriminators are tabulated in figs. 4.23a and 4.23b. On average, the SLAMs in both systems are filled almost equally (see table 4.2 below). For randomly connected SLAMs, the distribution is almost gaussian. For the clustered SLAMs, there are two maxima, one at one bit per SLAM which represents the SLAMs connected to low-activity n-tuples and the other maximum is that of a skew gaussian distribution. Also, in the clustered SLAM-PR, there are more SLAMs with more memory bits set and they represent the localized n-tuples in the high-activity regions.

This would explain why the memory of a PR using clustered SLAMs is relatively more difficult to fill. Only the SLAMs in the high-activity regions are filled, and for SLAMs which are partly filled to see new state of the n-tuple, the number of patterns required goes up exponentially as seen in subsection 3.4.3.

	Mean No. of Bits set per SLAM	Standard Deviation
Random SLAMs	8.38(52.3%)	0.62
Clustered SLAMs	8.44(52.7%)	0.72

TABLE 4.2

Mean and Standard Deviation of the No. of Bits Set in SLAMs

The optimum figure of recognition for the clustered SLAM-PR is only marginally greater than that of the basic SLAM-PR (fig. 4.22) and this improvement may be due to statistical variations. It is concluded that little is to be gained by the use of this technique. One positive result of this experiment is that the areas of important activities can be identified by the state of the SLAMs, and in future work one could reconnect all the SLAMs to those areas only. More is said about this in section 5.4.

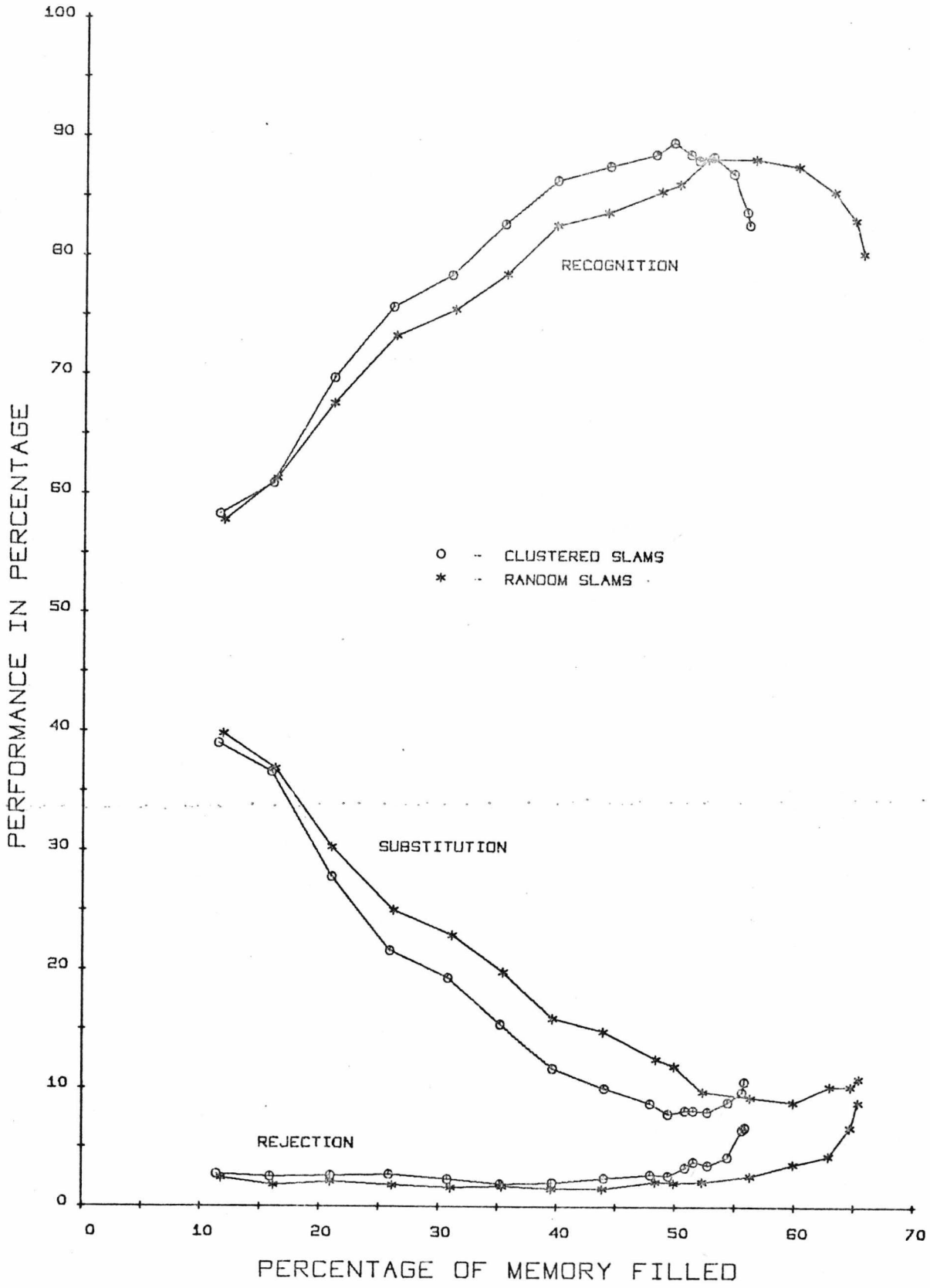

Fig. 4.22. Performance of Clustered & Basic SLAM-PRs.

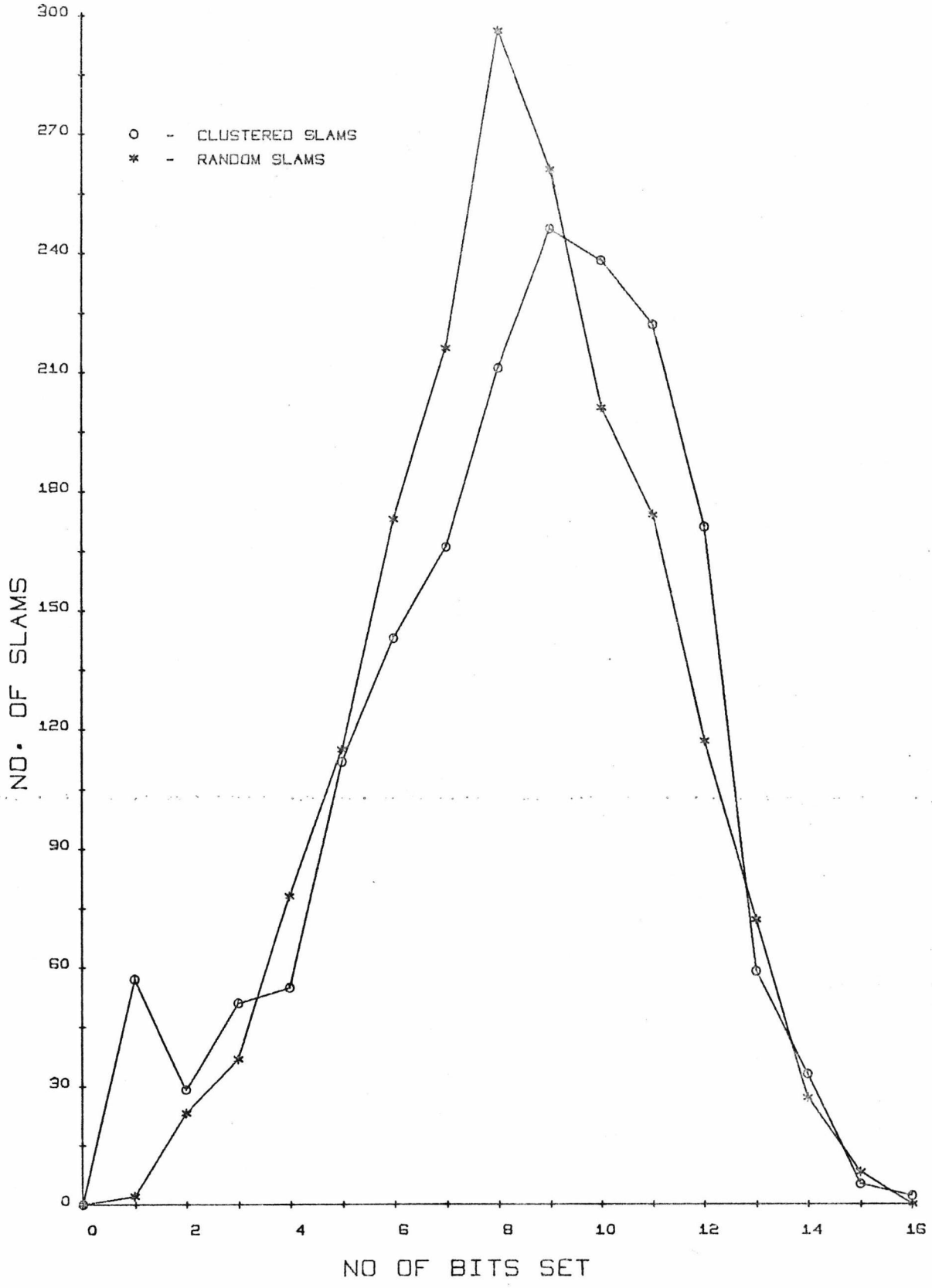

Fig. 4.23. Distribution of SLAMs with No. of Bits Set.

DISCRI-MINATOR FOR CLASS	NO. OF TRAINING PATTERNS	MEMORY FILLED (%)	NUMBER OF BITS SET															
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	100	30.2	2	20	26	45	28	30	8	9	5	3	2	1	1	0	0	0
2	30	56.0	0	0	0	1	4	11	25	34	40	26	20	10	8	1	0	0
3	35	55.0	0	0	0	0	9	17	26	36	32	18	14	18	9	1	0	0
4	31	55.1	0	0	1	3	9	14	24	29	32	28	22	9	3	4	2	0
5	33	55.1	0	0	0	4	10	13	25	41	24	14	21	13	12	2	1	0
6	48	55.0	0	1	1	7	10	22	15	31	22	19	18	16	14	4	0	0
7	100	50.6	0	0	5	9	20	23	21	36	17	9	13	12	7	5	3	0
8	22	55.7	0	0	0	1	5	8	24	39	39	29	19	11	4	1	0	0
9	38	55.3	0	0	1	4	10	18	27	17	26	28	23	15	9	2	0	0
0	42	55.0	0	2	3	4	10	17	21	24	24	27	22	12	5	7	2	0
OVERALL		52.3	2	23	37	78	115	173	216	296	261	201	174	117	72	27	8	0

Fig. 4.23a. Distribution of SLAM - 16 Modules with No. of Bits Set, for Individual Discriminators.

Each Discriminator Consists of 180 Randomly - Connected SLAMs.

DISCRI-MINATOR FOR CLASS	NO. OF TRAINING PATTERNS	MEMORY FILLED (%)	NUMBER OF BITS SET															
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	100	31.1	49	11	14	9	12	11	16	23	25	8	2	0	0	0	0	0
2	69	57.0	0	2	0	3	14	16	15	21	17	33	21	26	6	6	0	0
3	76	57.0	0	0	2	2	7	17	20	19	26	28	34	16	7	2	0	0
4	68	56.9	0	0	2	4	12	12	12	19	35	25	28	26	4	1	0	0
5	78	57.0	0	2	4	6	14	12	12	19	21	24	26	21	9	9	1	0
6	100	55.2	0	2	8	9	14	9	19	20	23	19	16	17	10	9	3	2
7	100	46.5	8	10	12	8	12	11	17	24	25	23	14	10	6	0	0	0
8	46	57.3	0	0	1	5	8	17	19	23	15	28	31	20	9	3	1	0
9	93	57.0	0	0	4	1	8	20	14	22	25	23	29	25	7	2	0	0
0	100	51.6	0	2	4	8	11	18	22	21	34	27	21	10	1	1	0	0
OVERALL		52.7	57	29	51	55	112	143	166	211	246	238	222	171	59	33	5	2

Fig. 4. 23b. Distribution of SLAM - 16 Modules with No. of Bits Set, for individual Discriminators.

Each Discriminator Consists of 180 Clustered SLAMs.

CHAPTER 5

DEVELOPMENTS SUGGESTED BY THE EXPERIMENTATION

5.1 Salient Characteristics of Basic and Output-Weighted SLAM-PRs

Some of the points which have been noticed in the simulation of SLAM-PRs (sections 4.5 and 4.7) are discussed here.

It has been found that the recognition rate increases with the size of the PR but tends to level off as the PR becomes large (fig. 4.9a). This diminishing return trend has been reported by Ullmann & Kidd (1969). That is, it is relatively easy to obtain a basic, medium recognition rate, but thereafter, any improvement becomes increasingly difficult. This characteristic is common to almost all pattern recognition systems and, in the present case led to an investigation such as output weighting in order to try to overcome the limitation. However, only a few of several such actions have been considered in this thesis. An obvious development is a revision of the way in which one chooses n-tuples, which is discussed in section 5.4.

As the memory of a basic or output-weighted SLAM-PR

is filled, the rate of substitution decreases rapidly. It will be argued that this may be explained partly in terms of the variation in the recognition rate and partly in terms of the increase in the rejection rate. The latter effect can easily be attributed to the fact that as more memory elements are set in each discriminator, the probability of more than one discriminator giving maximum response to a pattern increases, hence resulting in more rejections.

The Uniform-PR has been noticed to generate more rejections than the other output-weighted SLAM-PRs. In the Uniform-PR, if the same number of SLAMs in two or more discriminators respond positively and if that number is greater than in the remaining discriminators, then a rejection ensues. For the other output-weighted SLAM-PRs, a rejection does not necessarily follow if the same number of SLAMs in two discriminators respond positively. Nor is a rejection excluded when one discriminator has more SLAMs with positive outputs, but this is an unlikely event. It is the sum of the weighted positive responses which dictates the classification. Owing to the distribution of set memory bits among the SLAMs in a discriminator, (consequently the distribution of weighting values) the probability of two discriminators having equal sums is reduced.

It has also been found that the rate of recognition depends considerably on the level of memory filled; the recognition rate improves with training to an optimum and then deteriorates with more training. In particular, all output-weighted SLAM-PRs (except the Trough-PR) give an optimum performance at about the same level of memory filled. The optimum performance occurs between 30% and 60% of memory filled, depending on the size of the PR. As a consequence of this observation the experimental investigation described in the next section is carried out.

5.2 Hamming Distance Between Discriminator Memory Vectors

In this section, the variation of the recognition rate of a SLAM-PR with the average Hamming distance (HD) between the contents of the discriminator memories is considered. The memory of each n-input SLAM is taken as a vector. If S_{ij} is the i^{th} SLAM in the j^{th} discriminator, its associated vector is denoted as $\underline{S}_{ij}$ and consists of 2^n components.

$$\underline{S}_{ij} = [\, a_1, \ldots, a_k, \ldots, a_{2n}]$$

a_k = 1 or 0 depending on whether the k^{th} memory bit is set or not respectively.

Concatenating all the SLAM memory vector (SMVs) in a discriminator in a string results in what is referred to as a discriminator memory vector (DMV). The DMV for the j^{th} discriminator consisting of s SLAMs is

$$\underline{D}_j = [S_{1j}, S_{2j}, \ldots, S_{sj}]$$

The average HD between the DMVs in a PR is the sum of the HDs between all pairs of DMVs divided by the number of possible combinations.

A basic SLAM-PR (consisting of 45 SLAMs/discriminator) is simulated and the average HD between the DMVs at various levels of memory filled is determined and plotted in fig. 5.1. The recognition rate of the PR is also shown in fig. 5.1. The strong correlation between the two plots (fig. 5.1) and intuition suggest that the performance of a SLAM-PR can be improved by maximizing the average HD between the DMVs.

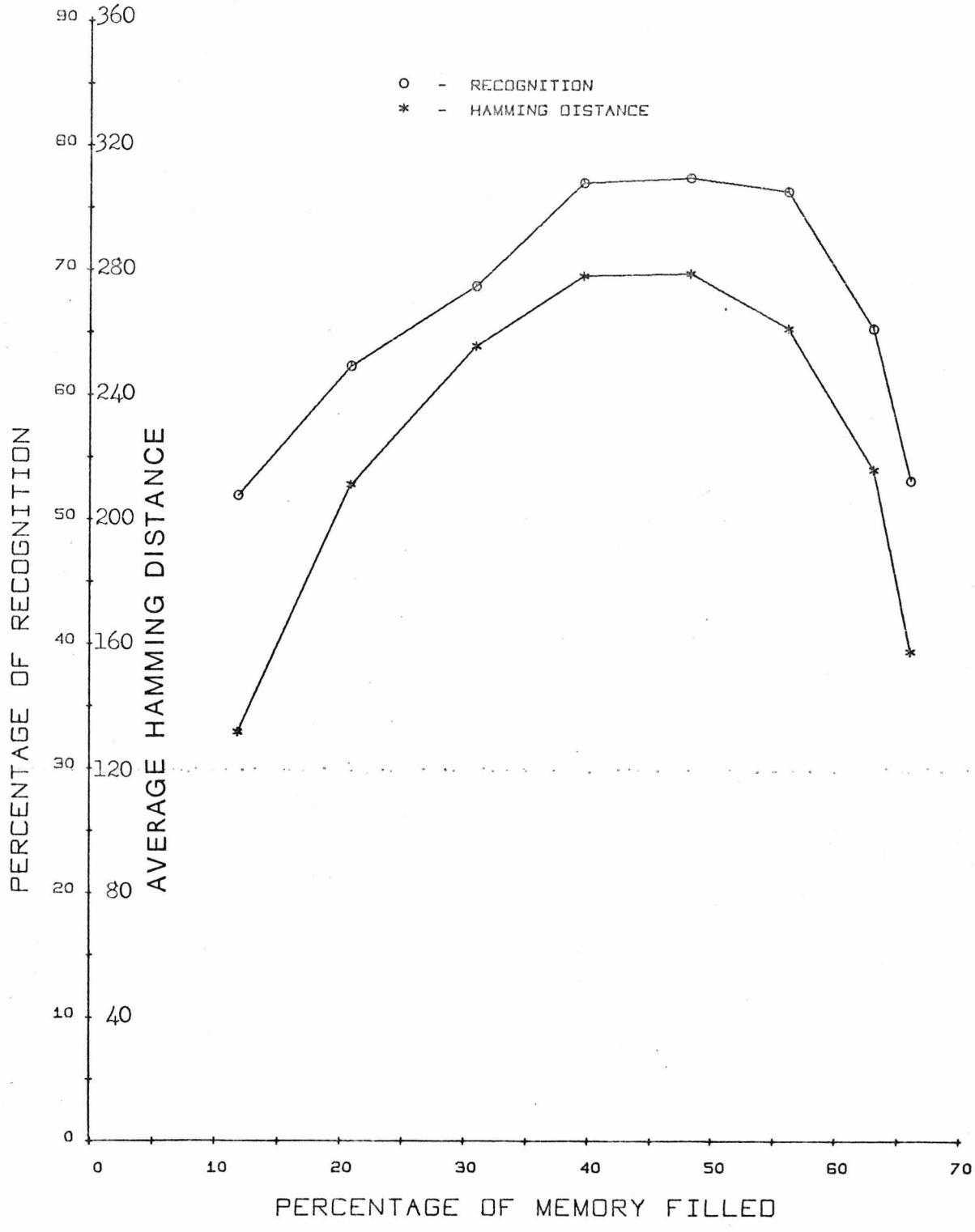

Fig. 5.1. Recn. Rate of Basic SLAM-PR & Average Hamming Distance between Discriminators against Memory Filled.

.5.3 Maximization of the Average HD Between DMVs

The criterion assumed during the training phase of a SLAM-PR is to keep the memories of the discriminators equally filled. That is, all the DMVs will contain an equal number of '1' components. It will be shown in what follows that by having the discriminators half-filled, the probability of having the greatest possible average HD between the DMVs is maximized.

Assuming that all the components in the DMVs have equal probability of being '1', the probability being equal to say q, then the probability of any component being '0' is (1-q). q would also be the likely ratio of '1' components to the number of components in the DMV.

Considering the i^{th} component of any two DMVs, the probability (P_i) of the two components being non-equivalent is given by

$$P_i = 2\{q(1-q)\}$$

since the probability of the first being,

'1' is q and the second being '0' is (1-q)

'0' is (1-q) and the second being '1' is q

The probability P_i is a maximum when q=0.5, i.e. when any component is equally likely to be a '1' or a '0'. Such a probability is equivalent to having the discriminator memories half-filled. This is also the condition for having the largest number of differing corresponding pairs of DMV components and hence the largest average HD between <u>any</u> two DMVs. Since no DMV is particularly favoured by the above argument, it follows that this is also the condition for maximum average HD between <u>all</u> the DMVs.

It should however, be pointed out that the assumption that all the components have equal probability of being '1' is not entirely true in actual practical cases. For instance, the states of an n-tuple lying in a corner of the retina do not have an equal probability of occurrence. Hence, this is only an approximate argument and leaves room for a more thorough theoretical treatment. Part of the inaccuracy of the above assumption leads to the conclusion that it is desirable to half-fill the discriminator memories and that the unbalance (due to corners, etc...) is due to the wrong choice of n-tuples.

5.4 Possible Technique for Choosing n-tuples

The maximization of average HD between the DMVs can be effected by maximizing the HD between corresponding SLAM memory vectors (SMVs). Using the same argument for the DMVs (section 5.3), the probability of obtaining the greatest average HD among SMVs is maximized when the SLAM memories are half-filled (optimal).

This, incidentally, could explain the odd result in the Trough-PR, whose performance is noticeably inferior than those of the remaining output-weighted SLAM-PRs (figs. 4.19a,....,e). The 'Trough' weighting scheme assigns zero weighting to the half-filled SLAMs.

One possible method of achieving the desired state of the SLAM memories is to have an abundant number of SLAMs per discriminator. Then after the training phase, the SLAMs which are optimally over-filled or under-filled can be removed.

Another method which is basically similar is to have a reconnecting procedure. The SLAMs are randomly connected and those which are not half-filled after the training phase are reconnected randomly. The procedure is repeated until an acceptable state of the

PR is reached. The repeating procedure effectively shifts the n-tuples connected to the SLAMs from low activity to higher activity regions, and vice versa.

Implicitly, the reconnecting process is a hill-climbing search for optimal n-tuples. The hill-climbing is done within the constraint of the initial random connection, thus if there is an absolute optimum connection the method does not guarantee finding it. Nevetheless, it may be of sufficient interest to find the local maxima within the above constraint.

5.5 Frequency of Occurrence of the States of n-tuples

In section 4.7, it is found that the variations in the performance for the different output-weighted SLAM-PRs are due to the different weighting schemes. One of the noticeable features is that the recognition rate curves for the 'Peak' and 'Trough' PRs consistently cross each other when the memory is about half-filled (figs. 4.19a,b,c,d,e). That is, for low levels of memory filled, a PR implementing the 'Trough' weighting scheme performs better than a Peak-PR.

This is due to the fact that the 'Trough' scheme,

at low levels of memory filled, has implicitly a higher weighting value for more frequently occurring states of its n-tuple. During the training phase, all the SLAMs in a discriminator see the same number of n-tuples (one n-tuple/pattern). Consequently, a SLAM which sees frequently occurring sets of inputs, is likely to have less memory bits set than the average for the discriminator, and if the overall level of memory filled is less than half, in the 'Trough' scheme the SLAMs with less bits set will be more heavily weighted.

The same implicit weighting effect would explain why the Peak-PR performs better when the memory is more than half-filled, and also why an Antiramp-PR generally performs better at any level of memory filled.

In section 4.5, it is noticed that for a given level of memory filled, the rate of substitution in a basic SLAM-PR is independent of the number of SLAMs (fig. 4.9b). This could have been due to the presence of 'rogue' patterns in the training sets and testing sets. A rogue pattern for a particular class has rogue states of n-tuples which occur very infrequently for the class. The SLAMs which have been trained on the rogue sets of inputs are more likely to cause

substitution error. Conversely, a rogue pattern in the testing set is more likely to be misclassified.

This underlines the weakness of most of the n-tuple methods for feature extraction, especially in problems that deal with unconstrained handwritten characters. The PR has to deal continually with new patterns as it is likely to be trained on patterns which occur very infrequently. It follows that one possible improvement, as already been shown by the discussion on the weighting schemes, is to attach more importance to features which occur more frequently.

5.6 Frequency-of-Occurrence-Dependent-Optimal SLAM

In this section a development to the SLAM is proposed, whereby the response depends on the frequency of occurrence of the individual state of the n-tuple. Moreover, the memory of the modified SLAM is such that it can only be filled to halfway or less, in the sense that it will respond positively to only half or less of the possible sets of inputs.

The outline of such a Frequency-of-Occurrence-

Dependent-Optimal SLAM (FOSLAM) is shown in fig. 5.2. The FOSLAM has a shift register related to each possible state of the n-tuple, as opposed to just one flip-flop in the SLAM. When presented with a set of inputs during the training phase, a bit '1' is forced into the associated shift register; so that for more frequently occurring sets of inputs, there will be more bits '1' in the pertaining registers.

To increase the efficiency of the registers, whenever the leftmost or bottom stage of all the registers contain '1', all the registers are shifted left by forcing a '0' in the rightmost or top stages, ensuring that there is always at least one register with the bottom stage containing '0'. The shifting left is achieved by means of a mono-stable element as shown.

Also, none of the registers is allowed to overflow, whenever a register is full, all the registers are shifted left, so that there is no bit '1' in any of the top stages.

During the training phase, assuming that the training patterns occur with some time-stationary first order probability and that they are not

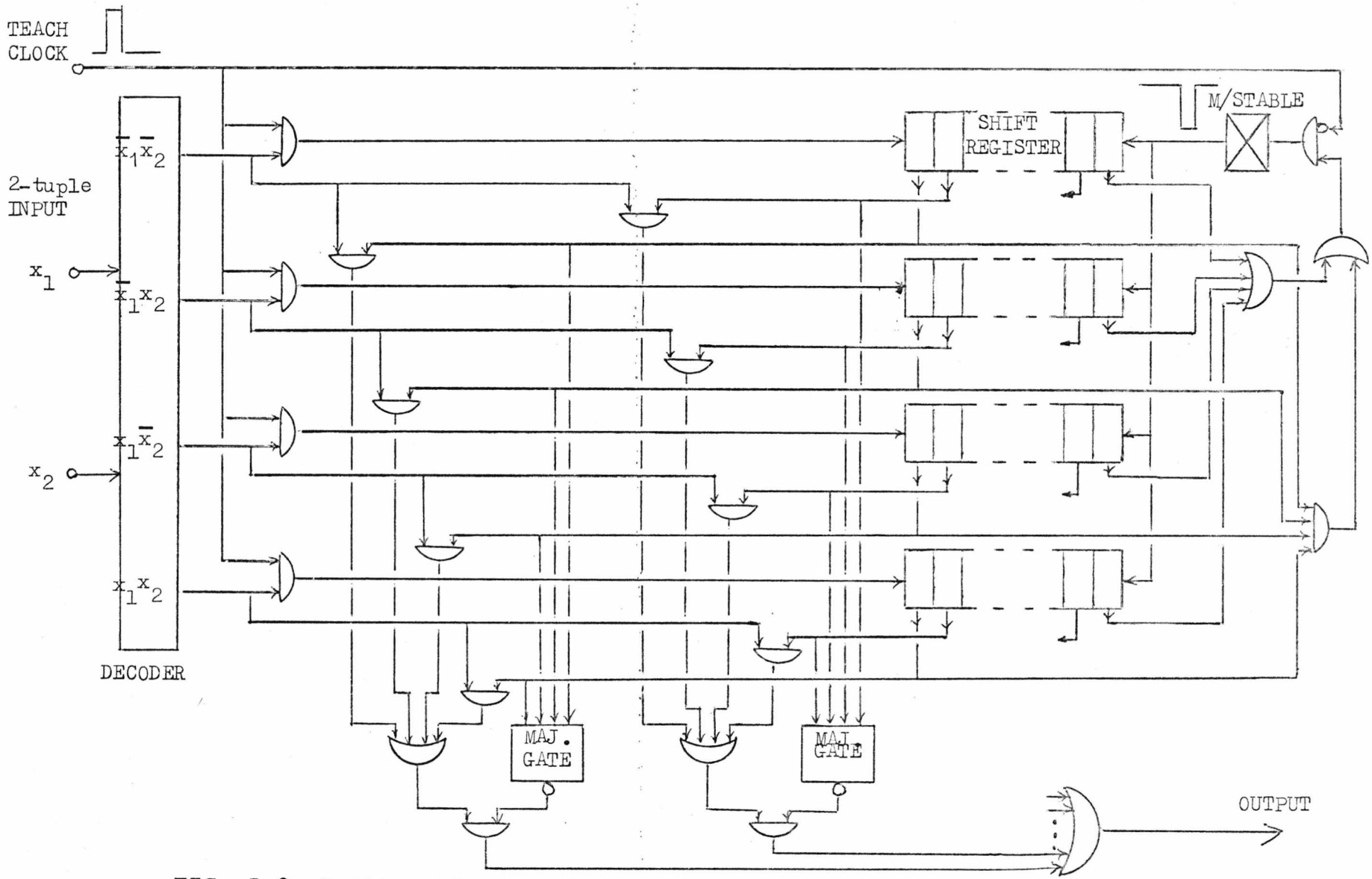

FIG. 5.2: Outline of a Frequency-of-Occurrence-Dependent-Optimal SLAM-4

'maliciously' arranged in any manner, the distribution of the bits '1' in the registers will, after some time, reach a dynamic equilibrium, when only the difference of occurrence is stored.

Each stage of the registers is fed into an inverted majority gate (only the logic for the first two stages of the registers is drawn in fig. 5.3). The output of an inverted majority gate will be '1' if half or less than half of the registers contain '1' in that particular stage. Therefore, during the recognition phase, only the inverted majority gates for the upper stages will output '1', and if the shift register addressed contains a '1' in the upper stages, then the response of the FOSLAM will be '1' or positive.

For hardware implementation, however, only one majority gate is required, an internal oscillator can generate clocking pulses which would cause all the registers to do a complete circular shift. If a positive output occurs during the circular shift, it sets a bistable element which stays set till the next pattern is presented. It is necessary to inhibit the monostable during the circular shift.

It is estimated that it is possible to have a

FOSLAM-8 with registers of 64 bits on a chip, or FOSLAM-16 with registers of 32 bits. The registers would occupy about half the chip, leaving the other half to accommodate the necessary logic.

Using FOSLAMs, only the more frequently occurring sets of inputs will give positive response, thus any rogue patterns in the training set will have less effect. Also larger training sets can be used, the capacity of the FOSLAMs being limited only by the length of the shift registers.

The FOSLAM can be modified so that it can be 'de-trained' on a set of inputs. Instead of forcing '1' in the register addressed, a '0' is forced down from the top, this is analogous to teaching a SLAM to respond negatively to a set of inputs. Owing to the integrating effect of the FOSLAM, the result of de-training is not as drastic as in a SLAM. A possible use of this technique is, for instance, when numerals '7' are very often classified in class '1' and vice versa. The discriminator for class '1' can be de-trained on numerals '7' and vice versa. The importance of the common features for the two classes is thus reduced and the features which represent the difference between the two classes is automatically accentuated.

CHAPTER 6

CONCLUSION

6.1 Quantitative Comparison

It is interesting to compare the results obtained here to those of Deutsch's recognition system (1968), as the same data has been used. Deutsch's system is adaptive, in the sense that the features (codes) are determined from a set of training patterns, but after the raw data has been transformed through a vast amount of preprocessing. The performance of his system and those of the various SLAM-PRs, simulated in this work, are shown in fig. 6.1 where the best performance figures for the systems are used.

It can be said that the recognition rate of a basic SLAM-PR, containing 160 SLAMs per discriminator, is comparable to that of the PR proposed by Deutsch. It must, however, be noted that whereas the preprocessing of the patterns here has been minimal and is feasible in parallel; Deutsch has performed elaborate sequential smoothing and thinning processes to reduce the thickness of the strokes. Furthermore, his recognition system is highly structured; the preprocessed pattern undergoes four coding stages, each being a sequential process, and the classification is

	Deutsch's	Basic SLAM-PR	Ramp-PR	Antiramp-PR	Entropy-PR	Peak-PR	Trough-PR	Clustered SLAM-PR
Recognition Rate (%)	87.5	88.0	89.7	89.6	90.2	87.2	79.2	89.6
Substitution Rate (%)	11.0	9.1	10.3	10.4	9.8	12.8	20.8	7.8
Rejection Rate (%)	1.5	2.9	0.0	0.0	0.0	0.0	0.0	2.6
No. of SLAMs/ Discriminator		160	160	160	160	160	160	180

Fig. 6.1. Performance of Various Recognition Systems.

effected by means of a tree-like algorithm which analyses the final code.

Owing to the sequential nature of the various operations, Deutsch's system would be costly to implement in hardware and the processing time may be prohibitive. In the present simulation of a basic SLAM-PR (fig. 3.2), containing 160 SLAMs per discriminator, the time taken to process one pattern is about ¼ second. A SLAM-PR, however, is more amenable to hardware realization and the feature extracting operations by the SLAMs would be done in parallel. The propagation delay through a SLAM-16 module which has already been manufactured (Glover & Aleksander, 1970) is 1μSec. One must add to this the time taken by the summing device and the maximum response detector which can be done with fast hardware.

6.2 Summary and Conclusion

This thesis has dealt with several pattern recognition systems using SLAMs. The hardware realization of such PRs is quite economic and their processing time would be fast because of the parallel operations of the SLAMs. They can be empirically

designed without making any assumptions about the probabilities of occurrence of the pattern classes and are not affected by changes in the probabilities.

A novel measure for training a SLAM-PR has been defined as the level of memory filled as different pattern classes are not of the same pattern density and degree of variability. In the training phase, the discriminators are filled as equally as possible independently of the number of training patterns from each class.

The performance of a SLAM-PR, which is applicable to pattern recognition problems in general, has been estimated on handwritten numeric characters and has been compared with that of a template-matching classifier. Using the recognition rate as a criterion for comparison, an optimally trained SLAM-PR has been found to perform better than the template-matching classifier for a given amount of memory storage.

It has been found that an increase in the number of SLAMs per discriminator generally improves the recognition rate. However, the trend of 'diminishing return' prevails as in most recognition systems. It has also been found that the performance reaches an optimum with memory filling (about one half) and then

deteriorates as the level of memory filled increases. Further experimentation shows that there is a strong correlation between the recognition rate and the average Hamming distance between the contents of the discriminator memories. It is therefore proposed to half-fill all the SLAMs by choice of n-tuples to increase the probability of obtaining a large average Hamming distance.

Ullmann (1969) has reported that given a constant number of training patterns, the recognition rate of a SLAM-PR reaches an optimum and then declines with increasing size of SLAMs (see section 4.3). This observation can now be interpreted in the light of the results obtained here. If SLAM-PRs are trained on the same number of patterns, the level of memory filled would shift relatively from high to low with increasing size of the SLAMs used, i.e. from over the optimum level for PRs using small SLAMs to below the optimum for PRs using large SLAMs.

PRs using SLAMs with weighted outputs have also been simulated. Various weighting schemes have been implemented and the variations in their performance indicate that improvement can be achieved by weighting more heavily the more frequently occurring features.

As a consequence of the observations in the experimentation, a possible development (FOSLAM) to the SLAM is suggested. A FOSLAM incorporates the characteristics of the SLAM, as regards speed and adaptivity; it will also respond positively to only the more frequently occurring features in the training patterns and to not more than half the positive sets of inputs.

It is hoped that further research will be carried along these lines, and that improved recognition will be achieved.

Digital pattern recognition is a complex and dynamic field, a field which is more or less contemporaneous with the emergence and growth of computer technology. There are still many oustanding problems, some of which are of a fundamental nature. Contributions towards the solution can be expected from both technological and physiological sources. With time a clearer understanding of the subject will emerge, meanwhile, the situation is best assessed by Harmon:

> "Digital pattern recognition has had a swift passage from birth to difficult and still clumsy adolescence. And with its less than twenty years of development instead of several billion perhaps our judgements should be tempered with charity."

APPENDIX 1. NOS. OF RANDOM PATTERNS TO FILL SLAM MEMORIES.

THE LIKELY NUMBERS OF PATTERNS, GIVEN THE PATTERN DENSITY, REQUIRED TO FILL THE MEMORIES OF SLAM-4, SLAM-8 & SLAM-16 HAVE BEEN ESTIMATED BY FORTRAN PROGRAMS. THE PROGRAM FOR SLAM-4 IS GIVEN BELOW. (IT IS ASSUMED THAT THE COMPONENTS '1' IN THE PATTERNS OCCUR RANDOMLY).

```
C
C    PROGRAM FOR SLAM-4
C
      DIMENSION P(4),IP(4),Q(4),X(4)
      WRITE (1,100)
  100 FORMAT (10H SLAM-4 P=)
C
C    TYPE IN PATTERN DENSITY - FORMAT F6.4
C
      READ (1,101) A
  101 FORMAT (F6.4)
      B = (1.0-A)
      P(1) = B*A
      P(2) = P(1)
      P(3) = A*A
      P(4) = B*B
C
C   O/P PROBS. OF OCC. FOR DIFFERENT STATES OF 2-TUPLE
C
      WRITE (1,102) P
  102 FORMAT (1H ,4F8.5)
      DO 103 I=1,4
      Q(I) = 0.
  103 IP(I) = I
    1 N = IP(1)
C
C    OUTPUT TRACING
C
      WRITE (1,104) N
  104 FORMAT (8H IP(1)= ,I2)
      N = IP(1)
      X(1) = P(N)
C
C    RUNNING SUM FOR PROB. OF SINGLE STATE OF 2-TUPLE
C
```

```
      Q(1) = Q(1)+Q(1)
      IF (IP(1)-4) 2,5,99
    2 N=IP(2)
      X(2) = X(1)*P(N)
C
C   RUNNING SUMS FOR JOINT PROBS.
C
      Q(2) = Q(2)+X(2)
      IF (IP(2)-4) 3,21,99
   21 IP(1) = IP(1)+1
      DO 22 J=2,4
   22 IP(J) = IP(1)+J-1
      GO TO 1
    3 N = IP(3)
      X(3) = X(2)*P(N)
      Q(3) = Q(3)+X(3)
      IF (IP(3)-4) 4,31,99
   31 IP(2) = IP(2)+1
      IP(3) = IP(2)+1
      GO TO 2
    4 N = IP(4)
      X(4) = X(3)*P(N)
      Q(4) = Q(4)+X(4)
      IP(3) = IP(3)+1
      GO TO 3
C
C    MULTIPLY BY NO. OF COMBINATIONS.
C
    5 DO 200 I=1,4
      DO 200 J=1,I
      AJ = FLOAT(J)
      Q(I) = Q(I)*AJ
  200 CONTINUE
C
C   OUTPUT PROBS.
C
  105 FORMAT (7H PROBS.,4(1H ,E14.4))
      WRITE (1,105) Q
      DO 106 I=1,4
      AI = FLOAT(I)
C
C   NO. OF LIKELY PATTERNS.
C
  106 Q(I) = AI/Q(I)
      WRITE (1,107) Q
  107 FORMAT (6H PATS.,4(1H ,E14.4))
   99 STOP
      END
```

APPENDIX 2

GENERATION OF PSEUDO-RANDOM NUMBERS

Pseudo-random numbers in the simulations have been generated by the recursive multiplicative expression:

$$R_{n+1} = (R_n.P) + 1 \ [\text{modulo } 2^{15}]$$

where R_n is the n^{th} random number

P is a multiplying factor.

The maximum possible sequence for the DDP-516, being a 16-bit word computer, is $(2^{15}-1)$; and the multiplying factors which give the maximum sequences are found to be

$$(2^{15}-3) - 4.I \qquad I \text{ is an integer.}$$

A random number (R_n') between a particular range is obtained by

$$R_n' = \left\{\frac{R_n}{2^{15}-1} \cdot (b-a)\right\} + a$$

where a is the lower limit

b is the upper limit.

REFERENCES

AKERS, Jr., S.B. and RUTTERS, B.H. (1964) : "The Use of Threshold Logic in Character Recognition" Proc. IEEE, Vol. 52, No. 8, pp. 931-938, August 1964.

ALBROW, R.C., ALEKSANDER, I. and NOBLE, P.J.W. (1967) : "An Adaptable Universal M.O.S.T. Monolith" Electronic Communicator 1967, 2.

ALEKSANDER, I. (1967) : "Adaptive Systems of Logic Networks and Binary Memories" Proc. Spring Joint Comp. Conf., 1967, p. 707.

ALEKSANDER, I. and ALBROW, R.C. (1968a) : "Pattern Recognition with Adaptive Logic Circuits" Proc. IEE-NPL Conf. on Pattern Recognition, July 1968.

ALEKSANDER, I. and ALBROW, R.C. (1968b) : "Adaptive Logic Circuits" The Computer Journal, Vol. 11, No. 1, May 1968.

ALEKSANDER, I. and ALBROW, R.C. (1968c) : "Micro-circuit Learning Nets: Some Tests with Hand-written Numerals" Electronic Letters, Vol. 4, No. 19, pp. 406-407, September 1968.

ALEKSANDER, I. and GLOVER, R.J. (1970) : "Digital Electronics Research Report"
No. 70/3/IA-RJG, University of Kent, Canterbury.

AMARI, S. (1967) : "A Theory of Adaptive Pattern Classifiers"
IEEE Trans. on Elect. Comp., Vol. EC-16, No. 3, pp. 299-307, June 1967.

ANDREWS, H.C. (1972) : "Introduction to Mathematical Techniques in Pattern Recognition"
New York, Wiley-Interscience, 1972.

BLEDSOE, W.W. and BROWNING, I. (1959) : "Pattern Recognition and Reading by Machine"
1959 Proc. of the Eastern Joint Computer Conf., pp. 225-232.

BLOCK, H.D. (1962) : "The Perceptron: a Model for Brain Functioning"
Rev. Mod. Phys., Vol. 34, No. 1, p. 123, 1962.

BOMBA, J.S. (1959) : "Alpha-Numeric Character Recognition Using Location Operations"
Proc. East. Joint Comp. Conf., 1959, p. 218.

CHIEN, Y.T. and FU, K.S. (1967) : "On Bayesian Learning and Stochastic Approximation"
IEEE Trans. on Systems Science and Cybernetics, Vol. SSC-3, No. 1, pp. 28-38, June 1967.

COOMBS, A.W.M. (1968) : "On the Systematic Construction of Features for Automatic Character Recognition"
Proc. IEE-NPL Conf. on Pattern Recognition, July 1968 (London:IEE).

COOMBS, A.W.M. (1972) : "The Special Case of Postcode Reading for the Automatic Sorting of Machine Printed Mail"
Proc. Inst. of Physics Conf. on Machine Perception of Patterns and Pictures, pp. 62-70, April 1972.

COVER, T.M. (1968) : "Estimation by the Nearest Neighbour Rule"
IEEE Trans. on Info. Theory, Vol. IT-14, No. 1, pp. 50-55, January 1968.

DEUTSCH, E.S. (1969) : "Character Pre-Processing and Recognition: A Pseudo-Topological Approach"
Ph.D. Thesis, University of London.

DOYLE, W. (1960) : "Recognition of Sloppy, Handwritten Characters"
Proc. Western Joint Comp. Conf., pp. 133-142, May 1960.

FU, K.S. and CHIEN, Y.T. (1967) : "Sequential Recognition Using a Nonparametric Ranking Procedure"
IEEE Trans. on Info. Theory, Vol. IT-13, No. 3, pp. 484-492, July 1967.

FU, K.S., CHIEN, Y.T. and CARDILLO, G.P. (1967) : "A Dynamic Programming Approach to Sequential Pattern Recognition"
IEEE Trans. on Elect. Comp., Vol. EC-16, No. 6, pp. 790-803, December 1967.

GENCHI, H. *et al*, (1968) : "Recognition of Handwritten Numerical Characters of Automatic Letter Sorting"
Proc. IEEE, Vol. 56, No. 8, pp. 1292-1301, August 1968.

HARMON, L.D. (1972) : "Automatic Recognition of Print and Script"
Proc. IEEE, Vol. 60, No. 10, pp. 1165-1176, October 1972.

HEBB, D. (1949) : "Organization of Behaviour"
Science Editions, Inc., New York, 1949.

HIGHLEYMAN, W.H. (1961) : "An Analog Method for Character Recognition"
IRE Trans. Elec. Comp., Vol. EC-10, pp. 501-512, September 1961.

HIGHLEYMAN, W.H. (1962) : "Linear Decision Functions, with Application to Pattern Recognition"
Proc. IRE, pp. 1501-1514, June 1962.

HU, M.K. (1962) : "Visual Pattern Recognition by Moment Invariants"
IRE Trans. on Info. Theory, Vol. IT-8, No. 2, pp. 179-187, February 1962.

LEVINE, M.D. (1969) : "Feature Extraction : A Survey"
Proc. IEEE, Vol. 57, No. 8, pp. 1391-1407, August 1969.

McCULLOCH, W. and PITTS, W. (1943) : "A Logical Calculus of the Ideas Immanent in Nervous Activity"
Bulletin of Math. Biophysics, Vol. 5, pp. 115-133, 1943.

MINSKY, M.M. (1963) : "Steps Toward Artificial Intelligence"
In Computers and Though, E.A. Feigenbaum and J. Feldman, Eds. New York: McGraw-Hill, 1963.

NAGY, G. (1968) : "State of the Art in Patttern Recognition"
Proc. IEEE, Vol. 56, No. 5, pp. 836-862, May 1968.

NAGY, G. (1969) : "Feature Extraction on Binary Patterns"
IEEE Trans. on Systems Science and Cybernetics, Vol. SSC-5, No. 4, pp. 273-278, October 1969.

NILSSON, N.J. (1965) : "Learning Machines"
New York: McGraw-Hill, 1965.

RAVIV, J. (1967) : "Decision Making in Markov Chains Applied to the Problem of Pattern Recognition"
IEEE Trans. Info. Theory, Vol. IT-13, No. 4, pp. 536-551, October 1967.

ROSENBLATT, F. (1960) : "Perceptron Experiments"
Proc. IRE, Vol. 48, No. 3, pp. 301-309, March 1960.

SARAGA, P. and WOOLLONS, D.J. (1968) : "The Design of Operators for Pattern Processing"
Proc. IEE-NPL Conf. on Pattern Recognition, July 1968 (London: IEE).

SELFRIDGE, O.G. (1959) : "Pandemonium : A Paradigm for Learning"
In Mechanization of Thought Processes, London: HMSO, 1959, pp. 511-535.

SELFRIDGE, O.G. and NEISSER, U. (1963) : "Pattern Recognition by Machine"
In Computer and Thought, E.A. Feigenbaum and J. Feldman, Eds. New York: McGraw-Hill, 1963.

TAYLOR, W.K. (1956) : "Electrical Simulation of some Nervous System Functional Activities"
In Information Theory, C. Cherry, Ed. Butterworths, Scientific Publications, p. 314, 1956.

TENERY, G. (1963) : "A Pattern Recognition Function of Integral Geometry"
IEEE Trans. on Military Electronics, Vol. MIL-7, No. 2 and No. 3, pp. 196-199, April-July 1963.

THOMAS, R.B. and KASSLER, M. (1967) : "Character Recognition in Context"
Information and Control, Vol. 10, No. 1, pp. 43-64, January 1967.

UHR, L. (1966) : "Pattern Recognition"
L. Uhr, Ed. New York: Wiley, 1966.

ULLMANN, J.R. (1968) : "A Simplification of the Problem of Choosing Features"
Proc. IEE-NPL Conf. on Pattern Recognition, July 1968.

ULLMANN, J.R. (1969) : "Experiments with the n-Tuple Method of Pattern Recognition"
IEEE Trans. on Comp., pp. 1135-1137, December 1969.

ULLMANN, J.R. and KIDD, P.A. (1969) : "Recognition Experiments with Typed Numerals from Envelopes in the Mail"
Pattern Recognition, Vol. 1, pp. 273-289, July 1969.

UNGER, S.H. (1959) : "Pattern Detection and Recognition"
Proc. IRE, Vol. 47, pp. 1737-1752, October 1959.

WIDROW, B. and HOFF, M.E. (1960) : "Adaptive Switching Circuits"
IRE Wescon Convention Record, Pt.4, August 1960.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor, Dr. I. Aleksander, for his invaluable help and encouragement. Thanks are also due to the directors of C. I. Data Centre, Farnborough, for the use of their incremental plotter.

I am indebted to the University of Kent, Canterbury for the award of a research studentship.

Finally, I would like to thank my wife, Dany, for her support and motivation which made this work possible.