This is to certify that the

dissertation entitled

Inference of Array Grammars
Under Noise and Distortion

presented by

Gautam Biswas

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Computer Science

Major professor

Date December 29, 1982

INFERENCE OF ARRAY GRAMMARS UNDER NOISE AND DISTORTION

By

Gautam Biswas

A DISSERTATION

Submitted to

Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

1982

ABSTRACT

INFERENCE OF ARRAY GRAMMARS
UNDER NOISE AND DISTORTION

By

Gautam Biswas

This thesis research deals with learning of the syntactic structure of two dimensional binary patterns that occur in image processing and pattern recognition. The development of a two dimensional inference scheme using a probabilistic array grammar that incorporates noise and distortion models is the primary contribution of this thesis. A two level inference scheme is proposed based on a block structured array grammar. The ideal pattern must be known, which implies knowledge of the nonterminal rewriting rules. The key step is the inference of intermediate grammars from probabilistic samples.

Complexity and discrepancy measures are the criteria for selecting a simple and 'best-fitting' grammar. A number of complexity and discrepancy measures for string grammars are reviewed and a new, computationally feasible discrepancy measure that involves both probabilistic and structural factors, is defined. The new measure is shown to possess a number of intuitively desirable properties. The discrepancy measure is extended to two dimensions to quantify the suitability of an inferred array language.

The inference scheme uses noise and distortion models to accommodate deviations from the ideal structure among observed patterns. An independent noise model and two intuitive distortion models are analyzed along with a powerful and comprehensive model based on Markov random fields. The problems of generating patterns and estimating parameters are studied. Robustness tests and parsing experiments demonstrate the versatility of the Markov random field inferential model. Not only is it applicable with a number of noise and distortion models, but it also allows small deviations from the ideal structure, thus providing a flexible inference scheme.

The inference scheme and the computation of the discrepancy measures require large amounts of computer time in the sequential mode of operation. SIMD and pipeline architectures are suggested to speedup computation significantly. Implementation of the proposed scheme with these architectures should open up a number of real time practical applications in industrial vision and robotics.

Dedicated to my father

the Late Professor Anil Bhushan Biswas

and my mother

Mrs. Anjali Biswas

ACKNOWLEDGEMENTS

I am indebted to my thesis advisor, Professor Richard C. Dubes, for his help, guidance and encouragement throughout the course of this thesis work. He has served as an excellent role model and has contributed greatly to the development of my research abilities. Special thanks go to Professor Raoul LePage, for his patient efforts and many helpful suggestions that contributed greatly to the development of the Markov random field model and to Professor Anil K. Jain, for his critical review of this thesis work.

I thank Professors Lewis Greenberg, Jacob Plotkin and Indranand Sinha for agreeing to serve on my Ph.D. guidance committee. Acknowledgements are also due to Dr. Erdal Panayirci and past and present graduate students of the Pattern Recognition and Image Processing Laboratory - Dr. George Cross, Dr. Steve Smith, Dr. James Coggins, Neal Wyse, Weichung Lin, Ardeshir Goshtashby, Rick Hoffman and Xiabo Li for their help and suggestions during the course of this work. I must not forget to mention Dr. and Mrs. Subinoy Chakravarty, who have made my stay in E. Lansing a very pleasant one, as well as my wife Sujata, for her (long distance) encouragement through those hectic final months of thesis writing.

The financial support provided by NSF grant ECS-8007106 and DER is also gratefully acknowledged.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Symbol	Usage
Λ	Empty matrix.
θ	Row Catenation
ϕ	Column Catenation
ϕ	Row or Column Catenation
a_i	First order parameter for interior points - Markov random field model
a_b	First order parameter for boundary points - Markov random field model.
$a_{i,i}$	Second order parameter for pairwise interaction among interior points - Markov random field model.
$a_{b,b}$	Second order parameter for pairwise interaction among boundary points - Markov random field model.
$a_{b,i}$	Second order parameter for pairwise interaction among interior-boundary points - Markov random field model.
a(.,.)	Weight factor for window function.
A	Parameter vector Markov random field model; $A = \langle a_i, a_b, a_{i,i}, a_{b,b}, a_{b,i} \rangle$
A_m	Specific subset of language L(G).
AG	Siromoney Array Grammar
b	Number of pixels in a block.
b_i	Number of interior pixels in block.
b_b	Number of boundary pixels in block.
$b_{...}$	General second parameter - Markov random field model.

B	Boundary template
B_q	Specific subset of language $L(G)$.
c	Number of pixels flipped in a block.
c_i	Number of internal pixels flipped in a block.
c_b	Number of boundary pixels flipped in a block.
C	Count vector; $C = \langle s_i, s_b, s_{ii}, s_{bb}, s_{bi} \rangle$
$C(G)$	Complexity of string grammar G.
$d(x,L)$	Proximity value between string x and language L.
d_*	Window proximity measure (rectangle or triangle).
d_{min}	Minimum proximity measure.
d_{rect}	Rectangular proximity measure.
d_{tri}	Triangular proximity measure.
$D_a(.,.)$	Absolute discrepancy measure.
$D(.,.)$	Structural discrepancy measure.
$D_{tot}(.,.)$	Total discrepancy measure.
D_{min}	Minimum structural discrepancy measure.
D_{rect}	Rectangular structural discrepancy measure.
D_{tri}	Triangular structural discrepancy measure.
Dist	Distance.
exp	exponential.
E	Expected value.
$f(x)$	Probability function.
$F_{.,.--,.}$	Interacting Potential.
G	Formal grammar.
$\underline{G}$	Class of formal grammars.

I	Set of terminal symbols or picture primitives.
IML	Intermediate Matrix Language.
K_c	Number of strings in (cardinality of) set L_c.
$\ell(x)$	length of string x.
L_A, L_B	Intermediate languages of array grammar.
$L_c(x)$	Window on set L centered at x.
L_S	Specific subset of array language.
L(G)	Language of grammar G.
$L(.)$	Likelihood function.
min	Minimum.
MG	Matrix Grammar.
MRF	Markov random field.
n(.)	Number of.
N	Pattern type.
Nb_k	Pixels in k^{th} order neighborhood of a particular pixel.
p	Probability of change
p_i	Probability of change for interior points.
p_b	Probability of change for boundary points.
p(x), q(x)	Probability of occurrence of string x.
p-AG	Probabilistic array grammar.
P	Set of production rules.
P_1	Nonterminal production rules of array grammar.
P_2	Intermediate production rules of array grammar.
Q	Potential function.
R	Sample covariance matrix.

R Finite set of probabilities.

R_1 Probabilities assigned to nonterminal rules.

R_2 Probabilities assigned to intermediate rules.

s_i Count of interior points that are 1's.

s_b Count of boundary points that are 1's.

s_{ii} Count of interior-interior pairs that are both 1's.

s_{bb} Count of boundary-boundary pairs that are both 1's.

s_{bi} Count of interior-boundary pairs that are both 1's.

S Sample (training) set.

S^+ Positive sample set.

S^- Negative sample set.

S Start symbol of grammar.

T Set of all possible realizations on a binary lattice.

T Change template.

u(.) Discrete probability distribution - Markov random field.

U[.,.] Discrete uniform distribution.

V Set of nonterminal symbols.

V_1 Set of nonterminal symbols of array grammar.

V_2 Set of intermediate symbols of array grammar.

V1,V2,--,V8 Coded vectors to convert intermediate language to string form.

W(.,.) Neighborhood function.

W1,W2,--,W8 Coded vectors to convert intermediate language to string form.

WLD Weighted Levenshtein Distance for strings.

WLD2 Weighted Levenshtein Distance for arrays.

x	Pixel value (0 or 1).
$\underline{X}$	Configuration of pixels on a lattice.
$z(i,j)$	Indicator function.
Z	Normalizing constant for probability distribution - Markov random field.

CHAPTER I

INTRODUCTION

Pattern Recognition techniques are major contributors to the field of machine intelligence and perception. They involve the description and analysis of data, including their categorization into identifiable classes by the extraction of significant features or attributes that are relevant to the problem being studied. The mathematical techniques available to solve pattern recognition problems can be broadly classified into the statistical, or decision-theoretic approach [31,43], and the structural, or syntactic approach [41,46,83]. The first involves measuring characteristic features on the input data and then classifying the resulting patterns by partitioning the feature space according to categories or pattern classes. An example is the use of Fourier descriptors [85,120] to describe the shape of objects from their outer boundaries. The structural approach involves the description of objects, or patterns, in terms of their parts (subpatterns), and the connectivity relations among these parts. An example is the use of primitive symbols (simple curves) and a formal grammar to describe different types of chromosomes [62,63].

Decision theoretic methods are based on the use of discriminant functions and are well suited for patterns that can be meaningfully

represented in vector form. However, there are applications, such as scene analysis and image segmentation, where the relationships among the various component parts (subunits) are nonnumerical in nature, but can be expressed in terms of discrete mathematical models such as formal grammars [18]. In such situations, the structural approach seems to be more appropriate for understanding and analyzing complex patterns than the statistical approach. Structural and syntactic pattern recognition techniques have been applied to a number of application areas including chromosome identification [62,63] character recognition [3], shape analysis [84], classification of speech [27,57], data compression [5], biomedical waveform analysis [2,12,107], fingerprint identification [75,90], image understanding [108], texture analysis [56,68], defect location [77,87], component recognition and placement [111,112], and testing of integrated circuits [82,121].

This thesis is primarily concerned with the description and analysis of two dimensional patterns using syntactic techniques. This involves two main steps - (i) the selection and extraction of primitives, and (ii) the inference or learning of connectivity relations among primitives in terms of formal grammars. In particular, we infer two dimensional formal grammar models with applications to the generation, description and recognition of two dimensional objects in mind. The patterns of interest here are binary images or pictures, so we choose pixel values 0 and 1 as our two basic picture primitives. Our main emphasis will be on the learning of a formal grammar to

describe a class of patterns. The study of the primitive selection problem is not covered here.

Two dimensional grammar models present many advantages in representing two dimensional patterns [98]. The relationship among the components (subpatterns) of a pattern can be directly represented in terms of two dimensional "concatenation" relations. A one dimensional scheme may require a number of preprocessing operations to reduce the two dimensional pattern to a one dimensional form. Traditional one dimensional schemes suffer inherent difficulties in modelling objects with disconnected parts or objects with holes. Two dimensional grammars are easily amenable to parallel processing [72,91,101]. Parallel forms simplify mathematical models for pattern representation [72] and significantly reduce parsing time, which can be exploited in developing very efficient classification schemes.

After proposing a formal mathematical model for describing two dimensional patterns based on array grammars, we will concentrate on the problems of learning or inferring a formal grammar for a class of patterns or pictures from a finite set of samples. Inference schemes generate a number of 'candidate' grammars from a set of training patterns. Complexity and discrepancy measures are criteria for selecting the simplest and 'best-fitting' grammar and are therefore studied in some detail.

A strong emphasis is placed on the learning of the structures of noisy and distorted patterns. Patterns observed in the sample set are different from ideal structures because of inherent limitations in the recording and digitization processes. Changes in the ideal representations of the pattern that are caused by phenomena independent of the underlying ideal structures are termed 'noise' processes, whereas those that depend on the underlying structure are called 'distortion' processes. A number of noise and distortion models are discussed, including one based on Markov random fields. We compare the robustness of the inference procedure under various noise models.

The grammatical scheme proposed in this thesis has several advantages. Automated industrial inspection and robot applications require well designed and very precise lighting systems to avoid problems like low contrast, shadows and extraneous detail, which increase the complexity of vision algorithms [47]. The use of a noise or distortion model in conjunction with an inference scheme when the lighting system is inadequate would incorporate some of these problems in the model description and keep the recognition scheme computationally feasible. Also, integrating the noise or distortion process with the learning scheme allows setting up discrimination or classification procedures which distinguish between patterns that have the same ideal structures, but have been subjected to different noise or distortion processes. Array grammars can generate patterns of

different sizes, but with the same proportion of length to width. This property can be used to advantage when an imaging device for robot vision does not maintain a fixed distance from the objects of interest. Conventional schemes, such as template matching, require an extra distance dependent scaling operation before recognition. Moreover, the integration of the noise and distortion processes into the array grammar model allows small deviations in the orientation of the pattern thus simplifying the registration problem. It may also help differentiate among two sets of parts such as two types of bolts that have similar shapes, but different proportions of length to width.

The remainder of this chapter briefly presents the grammatical inference problem and reviews techniques that have been used for two dimensional pattern representation. Lastly, the organization of this thesis is described and its main contributions are summarized.

1.1 Grammatical Inference

Grammatical inference algorithms obtain a formal grammar [28,52,95] from a finite set of training patterns, expressed as strings or arrays of primitives. The finite set of training patterns is generally referred to as the sample set, and consists of two disjoint subsets - the positive sample set, S^+, which contains patterns from the language generated by the grammar to be inferred, and the negative sample set, S^-, of patterns which do not belong to that language. The grammars $\{G_i\}$,

created by an inference algorithm should necessarily satisfy the following conditions :

$$S^+ \subset L(G_i) \text{ and } \quad S^- \subset \overline{L(G_i)},$$

where $L(G_i)$ is the language generated by grammar G_i and $\overline{L(G_i)}$ is the the complement of $L(G_i)$. Our inference schemes utilize only a positive sample set, S^+.

The sample set S^+ can be probabilistic. Each pattern in the training set has a probability and patterns with higher probability are observed more frequently [42]. The corresponding inferred probabilistic grammar associates a probability with each of its production rules to reflect the probability of occurrence.

Grammatical inference algorithms in the literature apply only to strings of symbols and are therefore one dimensional in nature. Inference of finite-state string grammars have been attacked analytically [9,42]. Many properties of context-free grammars are undecidable, so inference algorithms for such grammars require heuristic techniques, which are applicable to restricted subsets of that class of grammars. An example is the Crespi-Reghizzi algorithm for operator precedence grammars [24]. Inference algorithms for string grammars have been extended to other constructs, such as trees and webs, with the idea that they naturally represent higher dimensional patterns [14,46,65].

Enumerative and constructive methods of inference have been proposed in the literature. Enumerative techniques usually involve an exhaustive search through a large, specified class of grammars, such as finite-state grammmars, for 'candidate' grammars that best describe the samples [53,69,81,118]. Tree searching and state space methods, in conjunction with a cost function, are used to direct the search. Constructive techniques derive grammars from similarities among the syntactic structures in the sample set. Restriction to a particular class of grammars enables one to define specific rules to convert observed syntactic structures into production rules for the grammar. Rules can also be defined for adding, merging, and deleting productions based on some criterion, to obtain a more acceptable grammar. An example of a constructive inference technique is Cook's [22] 'hill climbing' approach to the inference of context free grammars.

Both methods require a criterion, or cost function, to determine the 'best' grammar from a set of candidate grammars. The two properties that are used to define a good grammar are its structural complexity, and its discrepancy or fit to the given sample set. It is trivial to infer a grammar whose language is identical to the given sample set. To be useful, an inference algorithm must infer a grammar which is a natural generalization of the training set, yet has minimal complexity.

Syntactic Pattern Recognition schemes are often used to discriminate among classes whose sample strings contain random errors, or when languages are not disjoint. The quality of recognition can be improved by inferring probabilistic or stochastic grammars (p-grammars) [42,110], where each production rule of the grammar has an associated probability. These probabilities indicate that some patterns occur more frequently than others, or certain substructures within the pattern are more favoured than other possible ones.

The general inference problem for string grammars has not been solved completely, and there are still a number of open questions, both in formal language theory and in the capability of general grammatical inference schemes. However, several techniques exist for inferring limited classes of grammars and we shall be using some of them to develop our two dimensional inference scheme.

1.2 Two Dimensional Picture Representation

Regarding a class of two dimensional pictures as a two dimensional language has led to the use of formal linguistics for picture description and generation. Pictures are seen as a concatenation of subpictures, which are in turn built up of still smaller parts, called the picture primitives. Some models express the relations among picture

primitives, either by string concatenation or by algebraic operators, whereas others generalize one dimensional string grammars to two dimensional forms.

1.2.1 Picture Description Languages

One of the first attempts to represent two dimensional objects in terms of primitive substructures was the Freeman [38] chain code. Among its drawbacks are extremely long and unwieldy strings for object description, susceptibility to noise, and difficulties encountered with the representation of disconnected curves. Narasimhan [78,79] used a syntactic description model for classes of pictures composed of line like elements, but his method does not seem very easily generalizable to complex pattern descriptions and inference schemes. Shaw [97] extended Freeman's simple left-right string concatenation to four different binary relations among primitives in his Picture Description Language (PDL), resulting in two dimensional properties being expressible in string form. This technique looks quite elegant, but certain simple geometric operations like reflection and rotation are difficult to describe and pictures are often not easily expressed in terms of primitives of the form required by PDL.

These methods reduce two dimensional objects to one dimensional string representations which makes it quite clumsy to describe disconnected parts, or configurations with embedded objects, or even

objects with holes. These limitations are overcome by structures such as trees [13,40,45] and graphs [20,86,92]. While trees can represent high dimensional patterns more naturally than strings, and can provide simpler parsing schemes, it can be shown that there is a direct correspondence between tree representations and context free grammars [46]. Therefore, they have the same difficulties as strings in complex pattern descriptions, and do not seem to present a very natural scheme for general two dimensional structures. Graph or web representations are the most general schemes available for the description of two dimensional patterns, but require extremely complicated parsing techniques which are presently not well developed [15]. The development of automatic learning and decision making procedures is extremely difficult for general two dimensional schemes. There are, however, proper subsets of web grammars called matrix and array grammars, that have well developed formal structures, are relatively easy to parse, and also seem to be well suited to picture generation and description. This thesis will concentrate on these structures.

1.2.2 Two Dimensional Grammars

Kirsh [58] gave the first example of a two dimensional array grammar for generating a class of labelled 45-degree right triangles. His technique is not easily generalizable to other classes of pictures. Yodokawa et. al. [119] attempted to construct an appropriate mathematical model for two dimensional grammars. These authors also demon-

strated the existence of two dimensional Turing-type acceptor automata.

Siromoney et al. [99] proposed a model of a two dimensional matrix grammar (G_1,G_2), which is essentially a two level string grammar. The horizontal string grammar, G_1, generates a string of length n from k possible (intermediate) symbols. Each symbol in this string serves as a start symbol for a vertical right-linear (regular) string grammar. Grammar G_2 is the union of disjoint grammars which generate string columns of size m, consisting of terminal symbols which are picture primitives. This two step process generates an mxn picture. (Details are presented in Appendix C). Wang [114] proposed a variation on Siromoney matrix grammars as a three tuple (G_1,G_2,M), where G_1 is a horizontal grammar defined as before, but G_2 is the union of k disjoint grammars which need not be right linear; $M = \{m_1,m_2,------,m_p\}$, where each m_i is a kxl matrix consisting of k production rules, one from each one of the grammars that make up G_2. In the matrices generated by these grammar forms, it is not possible to maintain a fixed ratio of length to breadth, since the column expressions are independent of the row expressions.

Rosenfeld and Milgram [74] defined another form of array grammar and acceptor automaton based on a Turing machine with a two dimensional tape. This scheme starts with an array of blank symbols and fills it with picture primitives using production rules which map subarrays into subarrays. The main problem with these grammars is the number of

restrictions placed on the production rules to obtain patterns in their desired shape, as explained in Chapter 2.

Siromoney et al. [100] described a theoretical array grammar model which is more powerful than the matrix grammar models, since it can generate picture models not representable by matrix grammars. Its rewriting rules use a two step block structured approach, and describe a picture model in a clear step by step procedure. These array grammars are also very natural generalizations of one dimensional string grammars and possess several properties of one dimensional grammars. We have based our structural model on the Siromoney array grammars. Its two step block structured approach has been exploited to describe a simple model that incorporates noisy and distorted patterns. A detailed formal description of array grammars is presented in the next chapter.

1.3 Organization and Contributions of the Thesis

Chapter 2 contains both a theoretical review of array grammars and a critical discussion of properties required by the learning procedures to be developed. This chapter provides a comprehensive review of grammatical inference literature, and the models that have been developed for the structural description of two dimensional patterns. The concept of a probabilistic array grammar is defined for the first time in Chapter 2.

Chapter 3 introduces the picture description model and the main inference scheme. Several problem areas, including the role of complexity and discrepancy measures, and the need for probabilistic noise and distortion models, are examined. The main contribution of this chapter is the development of a two dimensional learning scheme, and the integration of noise and distortion models into this scheme.

Chapter 4 describes complexity and discrepancy measures for string grammars. The emphasis is on the properties of these measures, and their significance in the grammatical inference framework. An important contribution of this chapter, is the definition of a new structural discrepancy measure that overcomes some of the limitations of the presently available measures. The concept of structural discrepancy for string languages is also extended to define a discrepancy measure for array languages.

Chapter 5 describes the theoretical background and the computational schemes used for three noise and distortion models. The computational aspects of the generation of patterns and the estimation of parameters for the proposed schemes are then studied. The main contribution of this chapter is the development of a general distortion model based on Markov random fields. This is the first application and implementation of nonhomogeneous Markov random fields to image processing.

Chapter 6 presents the results of several experiments applying the inference scheme in conjunction with the noise and distortion models to two different pattern classes. The two dimensional discrepancy measure is employed to assess the robustness of the inference scheme. Another contribution of this chapter is the discussion of computational matters and potential speedup using parallel schemes. Finally, Chapter 7 presents the conclusions of our study and suggests future research.

The original contributions of this thesis can be summarized as -

1) Development and study of a two dimensional model description and inference scheme that incorporates the description of noisy and distorted patterns.

2) The definition and use of a new discrepancy measure for string grammars that incorporates both probabilistic and structural differences.

3) The extension of the discrepancy measure for string gramars to a measure of suitability for array languages.

4) The adaptation of nonhomogeneous Markov random fields in defining a distortion process that is very general, and covers a number of other noise and distortion models.

CHAPTER II

ARRAY GRAMMARS

This chapter establishes the mathematical basis for the inference procedures proposed in this thesis. There are two different structural forms for array grammars, the Rosenfeld array grammars and the Siromoney array grammars. We first discuss the Rosenfeld form of array grammars and its limitations. Then we present a formal description of the Siromoney array grammar (AG), review definitions and notation and briefly describe array automata. Examples and illustrations clarify definitions and notation.

2.1 Rosenfeld Array Grammars

A Rosenfeld array grammar [74,91] contains production rules that replace one subarray of symbols, with at least one nonterminal symbol, by another subarray of symbols, which need not be of the same shape and size. These grammars produce arrays which are accepted by Turing machines with two dimensional tapes, where the primitive symbols are never rewritten. The initial array consists of a start symbol on one location of the tape and blank symbols on all others. The terminal array is made up of pattern primitives and blank symbols. Simpler

forms of these grammars could be defined by requiring the grammar to be monotonic, i.e., make the number of non blank symbols in the derivation sequences monotonically non decreasing. Rosenfeld [93] defined normal forms for these grammars and showed the equivalence of different types of array grammars. He also showed [91] that some of these grammars have parallel forms that are essentially the same as local digital picture processing functions.

Individual rows and columns of the host array in Rosenfeld array grammars must be shrunk or stretched by varying amounts to accommodate different rewriting rules. The implication of replacing one subarray by another of a different size and shape extends beyond the immediate vicinity of the subarray and causes relative displacements among rows and columns, and also overall changes in the outer boundary of the representation (Fig. 1). This effect, known as the shearing effect, makes it difficult to generalize properties of one dimensional grammars to two dimensions and causes difficulties in object description. Rosenfeld [91] circumvented this problem by introducing isotonic array grammars, which require the left and right sides of the productions to have the same shape. Cook and Wang [23,116,117] defined normal forms for these grammars and established the Chomsky hierarchy.

The shearing effect, and the steps taken to remedy it, make Rosenfeld array grammars rather cumbersome. These array grammars are not natural extensions of one dimensional grammars. The Siromoney and

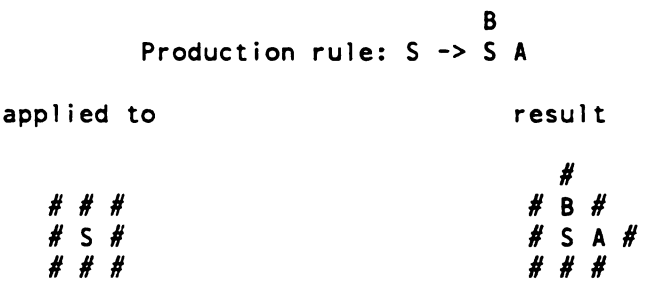


FIG. 1: Shearing Effect

Wang matrix grammars discussed in Chapter 1, though simple extensions of one dimensional grammars, lack the power to generate pictures of different sizes having the same proportion. For example, one cannot generate objects of size mxn such that m and n have a fixed ratio. The fact that the column grammars are all independent makes it difficult to express any relation between columns, as one goes down the rows. The Siromoney array grammars(AG) overcome the above difficulty and are free from the shearing effect. We base our structural model on the Siromoney array grammar form.

2.2 Siromoney Array Grammars

Structural models based on array grammars lend themselves very naturally to the generation of two dimensional patterns. Each cell represents a primitive structure of the pattern, and therefore a

terminal symbol of the array grammar. For example, if one considers a class of binary digitized pictures, the simplest set of primitives is {0,1}. An mxn rectangular picture array consists of mn samples from this set. As the number of primitives and the sizes of the arrays to be processed increase, one must resort to primitive selection techniques. We shall not be concerned with this problem, and assume we have two primitives, or terminal symbols for our array grammar.

The rewriting rules of Siromoney array grammars generalize the notion of string rewriting rules to array rewriting rules, with arrays of terminals replacing strings of terminals. In place of the conventional left to right concatenation, row and column catenation are defined, which establish two dimensional structure. Derivations are restricted by the conditions of row and column catenation; i.e. two matrices can be 'row catenated' only if they have the same number of columns and two matrices can be 'column catenated' only if they have the same number of rows. Rewriting rules can be regular(R), context-free(CF), or context-sensitive(CS). Appendix A clarifies the concept of row and column catenation of arrays, and defines some of the notation used in the formal definition of array grammars [60,100]. Probabilistic array grammars are introduced at the end of this section.

2.2.1 Formal Definition of Siromoney Array Grammars

The Siromoney array grammar AG is a four tuple,

$$AG = (V, I, P, S)$$

where V is the union of V_1, a finite set of nonterminal symbols, and V_2, a finite set of intermediate symbols;

I is a finite set of terminal symbols; $V \cap I = \phi$

$S \in V_1$ is the start symbol.

$P = P_1 \cup P_2 \cup P_3$; P_1 is a finite set of nonterminal rules, P_2 is a finite set of intermediate rules and P_3 is a finite set of terminal rules.

P_1 can be described as a finite set of ordered pairs, (u,v) written as $(u \rightarrow v)$, where $u,v \in (V_1 \cup V_2)^+$, or $(V_1 \cup V_2)_+$;

P_1 is context-sensitive if $\exists$ $(u,v) \in P_1$: $u = u_1 S_1 v_1$ and $v = u_1 a\, v_1$, where $S_1 \in V_1$, u_1, v_1, $a \in (V_1 \cup V_2)^+$ or $(V_1 \cup V_2)_+$.

P_1 is context-free if $\forall$ $(u,v) \in P_1$, $u \in V_1$, $v \in (V_1 \cup V_2)^+$, or $(V_1 \cup V_2)_+$.

Lastly, P_1 is regular if $\forall$ $(u,v) \in P_1$, $u \in V_1$, $v \in X \phi Y$ (ϕ is ϕ or θ), $X \in V_1$ and $Y \in V_2$ or vice versa.

P_2 is a set of ordered pairs (u,v), such that

$$u,v \in (V_2 \cup \{x_1,x_2, \text{-----},x_p\})^+, \text{ or}$$

$$u,v \in (V_2 \cup \{x_1,x_2,\text{-----},x_p\})_+;$$

where $x_1,x_2,\text{-----},x_p \in I^{++}$ are arrays of primitives with conformable rows and columns. The implication is that the intermediate rules involve only intermediates and a fixed and finite set of arrays in I^{++}. Further, each intermediate in V_2 generates a language, called the intermediate matrix language (IML), whose terminals are arrays with either the same number of rows or the same number of columns.

P_3 is a finite set of ordered pairs (u,v), $u \in (V_1 \cup V_2)$ and $v \in I^{++}$.

An array grammar is called an (XX:YY) array grammar, where XX denotes context-sensitive, context-free, or regular depending on whether P_1 is context-sensitive, context-free, or regular, and YY denotes context-sensitive, context-free, or regular if at least one intermediate language is context-sensitive, context-free, or regular, and all other languages are either the same or below it in the Chomsky hierarchy.

For convenience of notation we write the IML generated by $A \in V_2$ as,

$L_A = \{X \mid A \quad X \in (x_1,x_2,\text{----},x_p)^+, x_i \in I^{++}$ and all x_i have the same number of rows$\}$, or

$L_A = \{X \mid A \quad X \in (x_1,x_2,\text{----},x_p)_+, x_i \in I^{++}$ and all x_i have the

same number of columns}.

Beginning with the start symbol S, derivations proceed by applying the rules from P_1 successively until all nonterminals are replaced. This may involve the use of a P_3 rule if the symbol in the innermost parenthesis belongs to V_1. The operators ϕ and θ are not mutually associative, so proper parenthesization is required. Step two involves replacing each intermediate $A \in V_2$ by elements from L_A, subject to the conditions imposed by row and column catenation. The replacement starts from the innermost parenthesis and proceeds outward.

Since P_3 by itself does not play an important role in the derivation scheme, we modify the above definition of P and incorporate P_3 into P_1 and P_2. Therefore, P_1 can also have productions $u \rightarrow v$, where $u \in V_1$ and $v \in I^{++}$. An important fact to be noted is that derivations would come to a dead end if conditions for row and column catenation were not satisfied. Examples of array grammars are shown in Figs. 2 and 3.

This generation procedure requires that the nonterminal rules generate a set of blocks, catenated horizontally and vertically. This we call the block structure description of the picture. Filling in each block with primitive symbols using the intermediate rules P_2 creates rectangular arrays of primitive symbols. The set of arrays generated by an array grammar is called the array language, and the

NONTERMINAL = { S }; INTERMEDIATES = {A,B}; PRIMITIVES = {X,.}

NONTERMINAL REWRITING RULES: $S \rightarrow (A \theta S) \phi B.$

$S \rightarrow X$

INTERMEDIATE LANGUAGES: $L_A = \{ (.)^n \mid n>=1 \}$

$L_B = \{ (X)_n \mid n>=2 \}$

A REALIZATION OF SIZE 4:

(1) APPLY NONTERMINAL RULES -

$S \rightarrow (A \theta S) \phi B \rightarrow (A \theta ((A \theta S) \phi B)) \phi B$

$\rightarrow (A \theta ((A \theta ((A \theta S) \phi B)) \phi B)) \phi B$

$\rightarrow (A \theta ((A \theta ((A \theta ((A \theta S) \phi B)) \phi B)) \phi B)) \phi B$

(2) FILL INTO BLOCKS WITH INTERMEDIATE LANGUAGES -

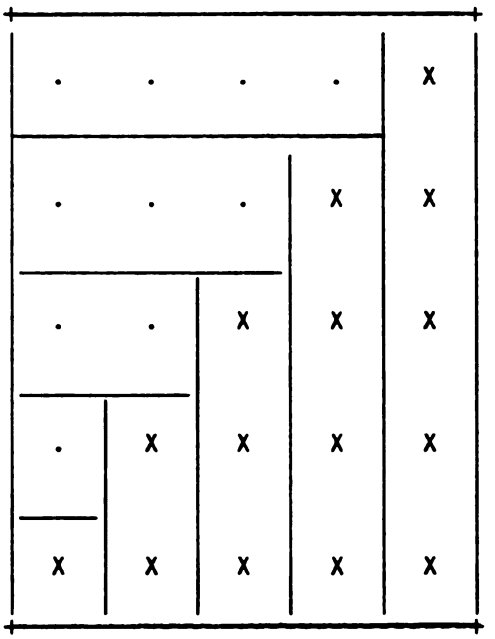


Fig. 2: Array Grammar - Isosceles Triangle

NONTERMINALS = { S ,T}; INTERMEDIATES = {A,B}; PRIMITIVES = {X,.}

NONTERMINAL RULES, P_1:

S -> (A ϕ T) θ B

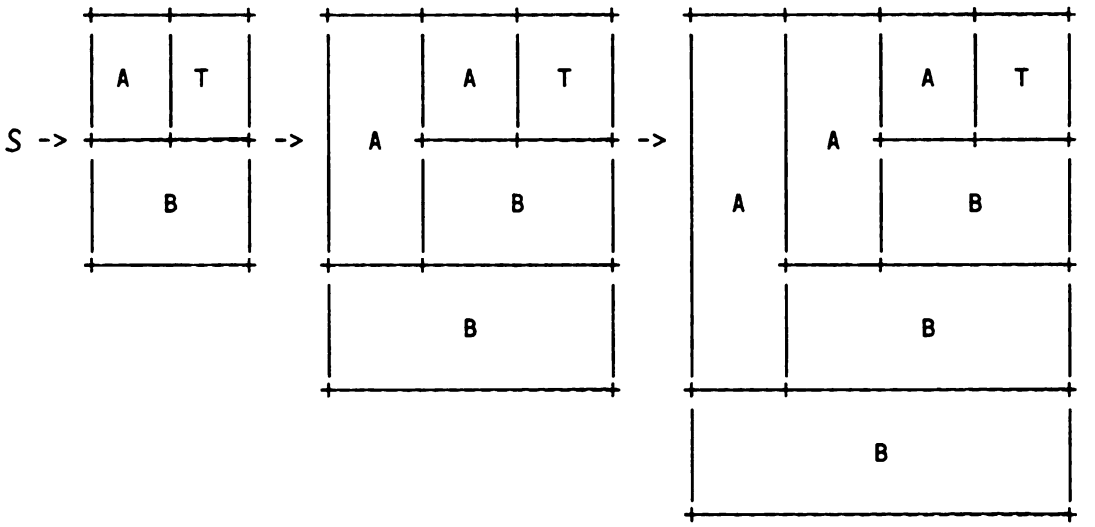
T -> (A ϕ T) θ B

T -> X X
X X

INTERMEDIATE LANGUAGES:

$L_A = \left\{ \begin{matrix} . \ . \ . \\ X \ X \ X \\ X \ X \ X \\ (. \ . \ .)_n \end{matrix} \;\middle|\; n >= 0 \right\}$ $L_B = \left\{ \begin{matrix} . \ . \ X \ X \\ . \ X \ X \ . \\ X \ X \ . \ . \end{matrix} \begin{pmatrix} . \\ . \\ . \end{pmatrix}^n \;\middle|\; n >= 1 \right\}$

A REALIZATION: (1) BLOCK STRUCTURE FROM NONTERMINAL RULES -



(2) APPLY INTERMEDIATE RULES -

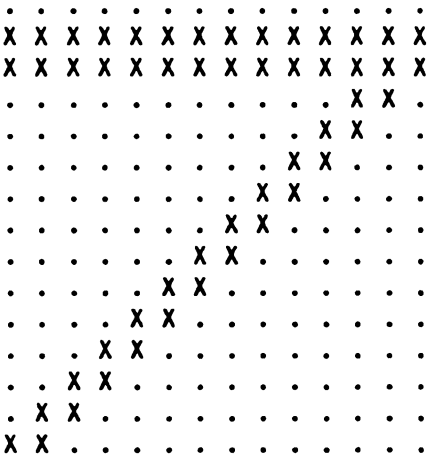


FIG. 3: Array Grammar for Numeral '7'

individual arrays are called patterns. In image processing applications these arrays represent digitized images.

2.2.2 Some useful properties

We now state some useful properties of array grammars which are stated as theorems and proved in Siromoney et al. [100].

I) The relations between various classes of matrix grammar languages (Siromoney) and array languages (Siromoney) is summarized in the following two dimensional scheme.

$$
\begin{array}{ccccccc}
\text{RML} & \subset & (\text{R:R})\,\text{AL} & \subset & (\text{R:CF})\,\text{AL} & \subset & (\text{R:CS})\,\text{AL} \\
\subset & & \subset & & \subset & & \subset \\
\text{CFML} & \subset & (\text{CF:R})\,\text{AL} & \subset & (\text{CF:CF})\,\text{AL} & \subset & (\text{CF:CS})\,\text{AL} \\
\subset & & \subset & & \subset & & \subset \\
\text{CSML} & \subset & (\text{CS:R})\,\text{AL} & \subset & (\text{CS:CF})\,\text{AL} & \subset & (\text{CS:CS})\,\text{AL}
\end{array}
$$

For example, all regular matrix languages (RML) are (regular,regular) (R:R) array languages and are also context free matrix languages (CFML).

This establishes the Chomsky hierarchy and implies that array grammars are more powerful than their corresponding matrix grammars. Since we are interested in developing inference schemes, we require that models use only regular and context-free grammars, both of which have well developed parsing techniques. Subsequent properties will be

stated only for regular and context-free grammars.

II) Let $\{M_n \mid n >= 1\}$ be an infinite sequence of matrices.

(1) For $n > 1$, if M_n is in one of the following forms-

$$M_n = (X \theta M_{n-1}) \phi Y, \text{ or } Y \phi (X \theta M_{n-1}),$$

$$\text{or } Y \phi (M_{n-1} \theta X), \text{ or } (M_{n-1} \theta X) \phi Y,$$

where X and Y are matrices from IML's L_X and L_Y and satisfy row and column catenation requirements, then $\{M_n\}$ can be generated by regular nonterminal rules.

(2) For $n >= 1$, if

$$M_n = X_1 \phi (Y_1 \theta M_{n-1} \theta Y_2) \phi X_2,$$

or

$$X_1 \theta (Y_1 \phi M_{n-1} \phi Y_2) \theta X_2,$$

where X_1, X_2, Y_1 and Y_2 are chosen from IML's L_{X_1}, L_{X_2}, L_{Y_1} and L_{Y_2}, subject to row and column catenation restrictions, and X_1 and X_2, or Y_1 and Y_2 are nonempty then $\{M_n\}$ can be generated by context-free nonterminal rules.

Property II illustrates the self embedding structure of (CF:XX)AG's analogous to the self embedding property of string context-free grammars. These two properties indicate the approach one may have to follow to develop a two dimensional inference scheme for the nonterminal rewriting rules.

It can be shown that all the (CF:XX)AL's are closed under row and column catenation. However, none of the (R:XX)AL's are closed under row or column catenation. All the (R:XX)AL's and the (CF:XX)AL's are closed under transpose, quarter- and half- turns, and reflections about both the base and rightmost verticals, indicating that there are relationships that one may use to parse patterns that are in the different configurations mentioned above, using the array grammar for the original pattern structure.

2.2.3 Probabilistic Array Grammars

A probabilistic array grammar is similar to a probabilistic string grammar [11,39]. The probabilistic array grammar (p-AG) is defined as a 5-tuple

$$\text{p-AG} = \{V, I, P, R, S\}$$

where V, I, P and S are defined as in Section 2.3.1 with $P = P_1 \cup P_2$.

$R = R_1 \cup R_2$; R_1 is a finite set of probabilities that are assigned in sequence to the nonterminal production rules in P_1 so that there is a probability associated with each production rule; similarly R_2 is a finite set of probabilities that are assigned in sequence to the set of intermediate rules.

The sum of the probabilities of all productions with the same left hand side must be 1. Assuming that all nonterminal and intermediate grammars are unambiguous, i.e., they have unique left most derivations, the probability function for an array language is defined as

$$f(x) = \prod_{k \in K} r_k^1 \prod_{l \in L} r_l^2 \quad \text{if } x \in L(AG)$$

$$= 0 \quad \text{if } x \notin L(AG)$$

where r^1 signifies probabilities of nonterminal production rules and r^2, probabilities of intermediate production rules; K and L are sets that represent the sequence of nonterminal and intermediate productions, respectively, that were used in deriving the element x. The p-AG is consistent if

$$\sum_{x \in L(AG)} f(x) = 1$$

2.3 Array Automata

Krithivasan and Siromoney [60] have defined array acceptors to accept languages generated by two dimensional array grammars as (AA:BB) array automata for (XX:YY)AG's; where AA and BB can be linear bounded, pushdown, or finite state depending on whether XX and YY are context-sensitive, context-free, or regular, respectively. The

automaton illustrates a generation scheme for array patterns and a two step parsing scheme that can be used for recognizing array languages. Step one consists of imposing the appropriate block structure on the pattern being tested, and step two checks if the contents of individual blocks are members of the corresponding intermediate matrix language. This approach is also exploited in the inference scheme described in the next chapter.

A formal definition of an array automaton and its implementation is given in Appendix B. We note that the automaton uses the sequential-parallel approach to processing. The first step is a sequential step by step process to subdivide the picture into blocks, and the second step is a parallel procedure where the automaton can work on each block independently and simultaneously.

2.4 Definition of Type and Block Number

In this section, we define the type of an array pattern, and block numbers, which label samples from particular intermediate languages.

Type: The pattern type is defined to be the number of times nonterminal production rules have to be applied to generate the required block structure for the pattern.

Block number: Block numbers are defined for the blocks that make up an intermediate language. All the blocks from an intermediate language

either have the same number of rows, or the same number of columns; therefore, they can be arranged in order of increasing number of rows if the number of columns is fixed, or in order of increasing number of columns, if the number of rows is fixed. Integer labels are assigned in sequence to the blocks of an intermediate language in order of their increasing size determined by their variable dimension. These labels are individually defined for each intermediate language of an array grammar.

For example, consider the nonterminal rule

$$S \rightarrow (A \; \phi \; S) \; \theta \; B$$

Applying the nonterminal rule four times we obtain the block structure shown in Fig. 4.

For the pattern of size 4, the four blocks of intermediate languages A and B have been labelled 1, 2, 3 and 4.

2.5 Summary

In this chapter we have formally defined array grammars, stated some of their useful properties and also described a type of sequential-parallel array automaton for accepting array languages. The examples suggest that these grammars provide a more powerful generative model than matrix grammars in terms of generating patterns of fixed

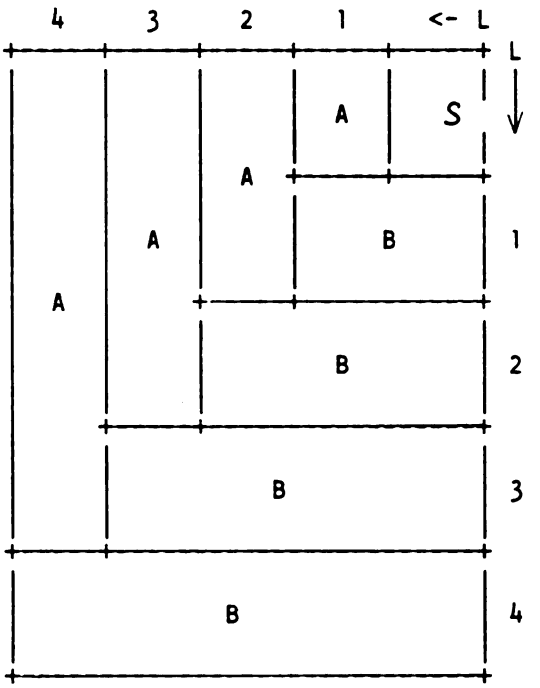


FIG. 4: Pattern Type and Block Number

proportions without the inconvenient shearing effects of Rosenfeld array grammars. The block structure approach to array grammars is exploited in defining a model for pattern description.

CHAPTER III

MODEL DESCRIPTION AND INFERENCE

This chapter develops the concept of two dimensional pattern description and learning based on the array grammar model and describes a scheme for inferring intermediate matrix grammars. Some interesting research questions that arise in the inference scheme are discussed. Techniques for computing block probabilities appears with the noise and distortion models in Chapter 5.

3.1 Pattern Description and Learning

The general inference problem can be stated as follows: Given a number of samples of rectangular arrays of various sizes from a class of patterns or images, infer an array grammar that describes the structure of this class of patterns. Section 2.3 suggests that this is a two step process. The first step is the inference of nonterminal rules, and the second is the inference of intermediate rules.

Prior information is required to set up the block structure, or the nonterminal rewriting rules, for the class of patterns. This leaves the learning problem to the inference of the intermediate matrix

grammars. Looking at this in another perspective, we are presented with a set of patterns for which ideal structures are known. The observable representations are distorted from the ideal structure primarily because of inherent limitations of the recording and digitization process. These pictures would not be accepted by an array automaton trained to recognize only the ideal structures of the class.

Uncertainty in string grammars has been modelled by probabilistic or stochastic grammars and error correcting parsing techniques. In probabilistic grammars [11,41,69], probabilities are assigned to each production rule and a sample is accepted in a particular class only if it is generated by the grammar representing the class with a high enough probability. Methods have been developed to infer p-grammars from a stochastic sample set, where a probability is given for each pattern in the sample set, or assigned according to a probabilistic model [42,53].

Another approach to handling uncertainty in one dimension has been to apply error correction to noisy strings and transform them to a form that belongs to the class of patterns being described. Three common types of corrupting errors are defined; insertion of extra symbols, deletion of existing symbols, and the change of one symbol to another. These are analogous to the three edit operations adopted to define distance measures between strings, such as the string edit or Weighted Levenshtein Distance (WLD) [64,80,113]. Error correcting parsing

techniques [1,44,67] develop proximity measures between a sample and a language. Acceptance in a particular class involves finding the string of minimum distance using a parsing algorithm (e.g. the Earley parser [33]). Lu and Fu [67] define a stochastic deformation model for patterns by extending a stochastic context-free grammar to a stochastic context-free error induced grammar. Liou [66] developed another type of heuristic inference procedure for regular and context-free grammars.

3.2 The Inference Scheme

One objective of this thesis is to incorporate distortions into the array grammar model, so that the automaton derived from the grammar accepts not only the ideal structure but also distored versions of the ideal structure. Section 2.2.3 defines probabilistic array grammars, and this concept is used to infer probabilistic grammars at the intermediate level.

The proposed inference scheme is outlined in Fig. 5. The first step is to impose a block structure on sample arrays using the nonterminal rewriting rules, which are assumed known, using an array automaton, as described in Sec. 2.3 and Appendix B. When the underlying noise or distortion model is also assumed to be known, Step 2 computes the probabilities of blocks extracted from the sample set. The inference problem is now reduced to the inference of p-grammars for each intermediate language from training sets made up of blocks with

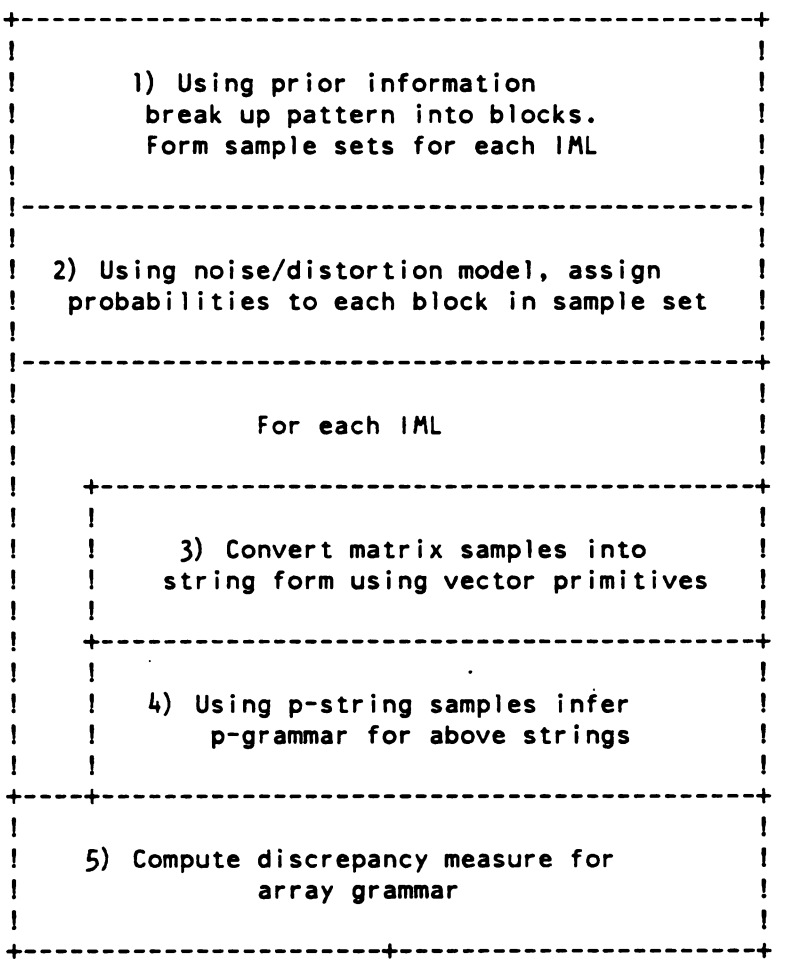


FIG. 5: Flowchart of Inference Scheme

associated probabilities.

Step 3 codes each block into a string form using the property that all matrices of an intermediate language either have the same number of rows or the same number of columns. For example, the blocks of IML L_A,

for the digit '7' in Fig. 6 have three columns whereas IML L_B has a fixed number of rows. When the number of columns is fixed, each row is coded as a single symbol. Thus rows represented by the pattern 'xxx' are coded as 'u' and the rows represented by '...' are coded as 'v'. Then

$$L_A = \{vu^2v^n \mid n >= 0\}.$$

IML L_A is thus reduced to a string form, with u and v as primitive symbols. The terminology 'vector primitive' for u and v means that they represent a row or column of primitive picture elements. Similarly, each column of a IML with a fixed number of rows is replaced by a coded vector primitive to reduce the elements of the language to a string form. As shown in Fig. 6,

$$L_B = \{pqrst^n \mid n >= 1\}.$$

If the number of rows or columns is k, the total number of possible vectors is 2^k.

Step 4 applies a standard grammatical inference algorithm to obtain a p-grammar. When a regular intermediate grammar model is used, the k-tail method of Biermann and Feldman [9] along with a maximum likelihood estimation scheme [42] will infer the probabilistic intermediate matrix grammars. The k-tail method produces a number of candidate grammars as the length of the tail, k, increases. The 'best' candidate grammar is selected with a criterion based on the complexity of the grammar and the discrepancy between the language of the grammar and the given sample set in Step 5. These complexity and discrepancy measures

$$L_A = \left\{ \begin{array}{c} \cdot\ \cdot\ \cdot \\ X\ X\ X \\ X\ X\ X \\ (X\ X\ X)_n \end{array} \;\middle|\; n >= 0 \right\}$$

Code X X X as u and . . . as v.

Then

$$L_A = \{ vu^2v^n \mid n >= 0 \}$$

$$L_B = \left\{ \begin{array}{c} \cdot\ \cdot\ X\ X \\ \cdot\ X\ X\ \cdot \\ X\ X\ \cdot\ \cdot \end{array} \begin{pmatrix} \cdot \\ \cdot \\ \cdot \end{pmatrix}^n \;\middle|\; n >= 1 \right\}$$

Code

$$\begin{array}{llllllll} \cdot & & \cdot & & X & & X & \cdot \\ \cdot & \text{as p,} & X & \text{as q,} & X & \text{as r,} & \cdot & \text{as s and} \quad \cdot \quad \text{as t.} \\ X & & X & & \cdot & & \cdot & \cdot \end{array}$$

Then

$$L_B = \{ pqrst^n \mid n>=1 \}$$

Fig. 6: Coding Intermediate Languages of '7' into String form.

are a key step in the inference scheme. Properties of different complexity and discrepancy measures are discussed in the next chapter.

The inference scheme poses a number of interesting research questions. The first involves the selection of mathematical noise and distortion models to describe the deviations in the observed patterns from the ideal pattern structures. Step 3 requires a moderate number

of vector primitives, so as to keep the inferred grammar simple. This gives rise to the question of how vector primitives may be optimally defined. Another research question is the establishment of a criterion for the suitability of the inferred array grammar.

A major contribution of this thesis is the development of mathematical noise and distortion models that describe deviations from the ideal pattern structure. These models are used to compute probabilities of individual blocks. A simple random noise model, two independent distortion models, and a distortion model based on Markov random fields are presented in Chapter 5 along with techniques for estimating the parameters of these models and computing the block probabilities. In Chapter 6, we study the robustness of the inference scheme to different noise and distortion models.

Once all the intermediate matrix grammars are inferred, we have an array grammar representation for the pattern class being considered. It is important to establish the suitability of this grammar to the pattern class, or how well it fits the training patterns. A discrepancy measure which compares the 'fit' between two array languages is developed in the next chapter.

3.2 Summary

This chapter describes a model for two dimensional patterns in terms of array grammars and proposes a learning scheme that infers probabilistic intermediate matrix grammars from noisy and distorted patterns, once the block structure description of the pattern class is known. The development of the two dimensional learning scheme and its use in conjunction with probabilistic noise and distortion models is a unique contribution of this thesis.

CHAPTER IV

COMPLEXITY AND DISCREPANCY MEASURES

A large number of grammars exist which generate languages that contain a given finite training set. A cost function is needed to evaluate the 'goodness' of candidate grammars generated by an inference scheme. This function is either intrinsic to the scheme or imposed on its outcome. At one extreme, the language of a candidate grammar could be the finite sample set itself. At the other extreme, the language includes all possible strings. The grammar that generates exactly the finite sample set tends to be more complex in structure than grammars that generate larger languages. Therefore, it seems natural that the 'goodness' of an inferred grammar be based on two measures - the grammatical complexity, which refers to the structural complexity of the inferred grammar, and the discrepancy, which measures fit between the language generated by the grammar and the given sample set.

On one hand, the inference scheme should pick the structurally simplest grammar as the solution grammar, but on the other hand, the language generated by the grammar should closely fit the given sample set and contain only strings that are logically implied by the training samples. The choice of the 'best' grammar will then depend on how much discrepancy one may allow in light of the complexity of the grammar.

Complexity and discrepancy measures have been defined for string grammars, but have not been generalized to other constructs such as tree or array grammars. In this chapter, we review the complexity and discrepancy measures cited in the literature, and pick a simple complexity measure that suits our inference scheme. A new discrepancy measure is defined based on dissimilarity measures among strings and its key properties are presented. The concept of the discrepancy measure for string languages is extended to array languages by defining a measure of fit between two arrays. This measure is then used to define the suitability of an inferred array grammar to the particular class of two dimensional patterns under consideration.

4.1 Complexity Measures

Complexity measures in one dimension are based on the size of a grammar or on information theory. Size complexity measures count the number of nonterminals, terminals and productions of a grammar, while information theoretic measures are based on probabilistic models. In the first two parts of this section we briefly review size and information theoretic complexity measures. In the last part, we compare these measures and select one for our inference scheme, based on its adequacy to effectively order inferred grammars by their complexity, and its computational complexity.

4.1.1 Size Complexity Measures

Size complexity for string grammars can be dynamic or static. Dynamic measures [51] deal with the notion of the 'speed' with which a grammar generates its language, and are therefore directly tied to the number of steps an automaton would take to recognize a string. This concept has been used for language parsing in compiler theory. Static measures [10,35] have been more widely used in the grammatical inference framework, because they are relatively easy to compute. They are usually based on the number of nonterminals, production rules and/or the sizes of production rules.

Gruska [48], Blum [10] and Wharton [118] defined complexity measures by mapping a class of grammars $\underline{G}$, into the nonnegative integers. Gruska's definition allows intuitively unacceptable mappings, such as constant mappings [118]. Blum's measure, defined on $\underline{G}$, and a fixed terminal set, I, is based on two important axioms; (i) there are a finite number of equivalence classes of grammars of any given complexity, and (ii) there is an effective procedure which determines for any positive integer, c, which grammars are of complexity c. For example, a complexity measure C, for the class of context free grammars in Chomsky normal form is,

$$C(G) = n(V) + n(P) + n(I)$$

where, $n(V)$ is the number of nonterminals, $n(I)$ is the number of

terminal symbols and n(P) is the number of production rules. A complexity measure valid for one class of grammars may not be valid for a larger class of grammars. The complexity measure defined above cannot apply to the general class of context free grammars, because the finiteness of the complexity classes depends on the fact that the maximum length of the right hand side of a production in the Chomsky normal form is two.

Wharton [118] remedied the above problem by introducing the maximum length of the right hand side of a production into the complexity measure. Feldman [35] also defined a general complexity measure to be expressible in terms of an intrinsic complexity which is EAO (effectively approximately ordered) by the number of symbols required to write the grammar, and a derivational complexity. This complexity function is a computable, unbounded, increasing function of each of its two arguments. Feldman's measure is quite restrictive in the sense that one measure, length, is used as a cannonical measure, in terms of which all other measures are defined. Blum's definition of complexity is more general, because it does not require a specific cannonical measure. All the size complexity measures discussed above, are based on intuitively simple concepts, and are directly linked to the structural size of grammars.

4.1.2 Information Theoretic Complexity Measures

Information theoretic complexity measures are based on probabilistic grammar models. The term 'information' comes from Information Theory, and represents the decrease in uncertainty that comes about with the occurrence of an event in a stochastic process.

Cook [21] defined a complexity measure for grammars as the information required to specify the grammar, or the sum of the complexities of its productions. The complexity of a production is determined by its right hand side and the complexity of a string is computed in terms of a grammar-grammar. This measure assumes that the elements of productions are statistically independent, which may not be realistic. For example, context free grammars exhibit similarity among productions. A way around this is to introduce conditional formulation.

Other approaches have been suggested by Horning [53], who derived a complexity measure from Bayesian theory, and Feldman et al. [34] who defined the derivational complexity of the sample set relative to the grammar G. The derivational complexity is closely related to the probability of the string being generated by the grammar. The less probable the string, the higher its derivational complexity. Therefore, this measure is similar to Horning's because it maximizes the probability of occurrence of the sample set by the grammar.

Complexity measures have also been defined [19,61] in terms of the generative capacity of grammars, called the entropy or channel capacity of the grammar, and depend directly on the number of words generated by the grammar. Chomsky and Miller [19] formulated the information capacity of regular grammars in a manner similar to the channel capacity concept of Shannon [96]. Kuich [61] developed similar results for context free grammars and defined entropy in terms of the structure generating function for a grammar.

All the information theoretic measures relate the complexity of a grammar to the information required to specify the grammar, in terms of the number of production rules and the number of symbols used. Whereas size measures are directly based on counts and are computationally much simpler, these measures use probabilistic concepts to derive information theoretic quantities, resulting in measures that are computationally more complex. The choice of a complexity measure is highly dependent on the problem at hand. Therefore, it is not possible to define general criteria for a best complexity measure.

4.1.3 Choice of Complexity Measure

This section describes our choice for the complexity measure to be used in conjunction with the inference scheme for intermediate string

grammars. The measure is easy to compute, but permits quantification of the differences in complexity among grammars generated by the k-tail inference scheme described in the last chapter.

Size complexity measures, such as Blum's and Wharton's, are very easy to compute, as opposed to those based on information theory. Moreover, Cook's and Feldman's measures are tailored towards context free grammars and are quite redundant for regular grammars. Horning's measure requires the specification of the intrinsic probability of a grammar, which is actually the a priori probability of that grammar in the given class of grammars. This parameter is very difficult to define in our application.

For these reasons, we choose the size complexity measure defined by Blum. We define the set $\underline{G}$ (Sec. 4.1.1), as the set of regular grammars generated by the k-tail inference scheme for different tail lengths. A complexity measure for regular grammars requires only the number of production rules. The complexity of the inferred grammars then increases monotonically with increasing tail lengths.

4.2 Discrepancy Measures

Discrepancy measures for string grammars evaluate the match between the language generated by a grammar and the given training set of patterns. These measures can use the differences in the probability

distributions of the elements in the sample set and those generated by the grammar or they can be based on structural differences between the sample set and the language.

The discrepancy between two probabilistic languages L_1 and L_2, can be made up of three parts corresponding to the subsets, $L_1 \cap L_2$, $L_1 - L_2$ and $L_2 - L_1$. Languages L_1 and L_2 are said to fit closely if the strings in L_1, but not in L_2, are structurally similar to the strings in L_2; so also for strings in L_2 which are not present in L_1. Various measures have been proposed to quantify these ideas, such as the Weighted Levenshtein Distance (WLD) [80,113] and the similarity measures of Findler [36]. Liou [66] has defined variations of the WLD and studied their properties. We use the WLD as the measure of dissimilarity among strings.

In the inference framework, language L_2 is the finite sample set, S. The discrepancy measure should reflect how closely the language of an inferred grammar fits S, and whether it is a natural generalization of S. Since $L(G) \supset S$, the discrepancy measure should depend on probabilistic differences for strings belonging to S, and structural differences on the set $L(G) - S$. The discrepancy or difference of fit, is a form of proximity measure between two sets of strings. Therefore, it would be desirable that they satisfy the metric properties, analogous to distance measures in metric space. Measures of fit should be bounded and convergent; otherwise the discrepancy might increase

without bound for languages L(G) that are infinite generalizations of S. Convergence of the measure also allows a comparison of individual discrepancy values against an upper bound and makes the discrepancy measure computationally feasible. In this section, we review existing probabilistic and structural discrepancy measures and motivate discussion for a new discrepancy measure.

4.2.1 Probabilistic Discrepancy Measures

Maryanski [69] defined several discrepancy measures for probabilistic languages based on the absolute difference or the square difference between probabilities. Given two probabilistic languages L_1 and L_2 with word probabilities p(x) and q(x) respectively, he defined six difference measures which are not metrics. An example is the absolute difference measure

$$D_a(L_1,L_2) = \sum_{x \in L_1} |p(x) - q(x)| \neq D_a(L_2,L_1)$$

One difficulty of these measures is that as L_1 increases monotonically in size, the distance between L_1 and L_2 increases monotonically, so a 'best' grammar is not easily selected. Generalizations resulting in infinite languages would be rejected in favour of finite generalizations.

Cook [21] defined a discrepancy measure between a finite training set of samples and a language, which he interpreted as the information lost or gained in going from the string probabilities in the sample set to those in L(G), even though this measure has no basis in information theory. Another problem with this measure is that there is no guarantee that it will converge for infinite languages. Liou [66] defined a procedure for limiting the number of strings to be considered in computing the sum for Cook's discrepancy measure, while trying to keep the information loss as low as possible. This procedure is still computationally expensive, and does not directly reflect structural differences among strings.

4.2.2 Structural Discrepancy Measures

Structural differences between the training set and an inferred language can be evaluated in terms of a distance measure between two strings, such as the string-edit, or the Weighted Levenshtein Distance (WLD) [64,80,113]. A string that belongs to both the sample set and the language of the grammar being evaluated, would not contribute to the discrepancy measure. Otherwise if $x \in L(G)$ but $x \notin S$, then $d(x) = \min \{dist(x,y)\}$. A general discrepancy measure could be defined as,

$$D[L(G),S] = \sum_{x \in L(G)} w(x)\ d(x). \qquad \text{---(1)}$$

where $\{w(x)\}$ is a sequence of weights that corresponds to the strings of the language $L(G)$.

This idea was used by Wharton [118] to define metrics on a class of languages over a finite terminal vocabulary to measure the difference between two strings. He defined a discrete metric, which is used in exact language identification but is not useful for inference.

The set $L(G)-S$ is countably infinite, so evaluation of (1) may not be computationally feasible. There is no guarantee that the sum converges. Therefore, this measure, too, is not very useful for our purposes.

4.2.3 Choice of Discrepancy Measure

Maryanski's discrepancy measures, though computationally simple, ignore structural differences and Wharton's weight measure ignores probabilistic differences. Cook's measure incorporates both to some extent but has the drawbacks discussed above. Therefore, we have decided to define a new discrepancy measure, which takes into consideration both structural and probabilistic differences.

4.3 New Discrepancy Measure

In this section, we propose a new discrepancy measure for string grammars that possesses a number of desirable properties. We begin with a formal definition of the measure, study its properties in relation to the inference scheme, and also discuss computational matters. A probabilistic difference measure, such as the absolute difference measure in Section 4.2.1, adequately measures the discrepancy for S. In Section 4.3.1 we define a new discrepancy measure for $L(G)$-S, $D(L(G),S)$, which is a special case of the general structural discrepancy measure (1) used by Wharton. The overall discrepancy between $L(G)$ and S is defined as the following linear combination of the above two measures.

$$D_{tot}(L(G),S) = b_1 D_a(S,L(G)) + b_2 D(L(G),S) \qquad --(2)$$

where b_1 and b_2 are positive real numbers which determine the relative importance of each discrepancy in the total discrepancy measure. The choice of b_1 and b_2 depends on the application at hand, and will be discussed in greater detail in Chapter 6.

4.3.1 Definition

The structural discrepancy measure, $D(L(G),S)$, is based on structural differences between probabilistic languages. Two languages L_1 and L_2 are defined to be structurally equivalent, iff

$$\forall x : x \in L_1 \Leftrightarrow x \in L_2.$$

The structural difference between L_1 and L_2 is expressed as,

$$D(L_1,L_2) = \sum_{x \in L_1} p(x)\ d(x,L_2) + \sum_{y \in L_2} q(y)\ d(y,L_1) \qquad --(3)$$

where $p(x)$ is the probability of $x \in L_1$ and $q(y)$ is the probability of $y \in L_2$; $d(z,L)$ is some measure of proximity between string z and language L. Equation (3) can be rewritten as

$$D(L_1,L_2) = E[d(X,L_2)] + E[d(Y,L_1)]$$

where E denotes the expected value implied in (3), and X and Y are random variables representing strings from languages L_1 and L_2 respectively.

The discrepancy measure $D(L_1,L_2)$ is a proximity or dissimilarity measure between the two languages L_1 and L_2. Anderberg [4] provides a thorough review of such measures. They are normally required to satisfy the following properties :

(I) $D(L_1,L_2) >= 0$, with equality if and only if, L_1 and L_2 are structurally equivalent,

(II) $D(L_1,L_2) = D(L_2,L_1)$.

A third condition, the triangle inequality, i.e.

$D(L_1,L_2) <= D(L_1,L_3) + D(L_2,L_3)$

would result in the discrepancy measure having metric properties. However, we have not found a convenient and useful way of defining $D(L_1,L_2)$ to make it satisfy the triangle inequality. Condition (II) is always satisfied irrespective of the way we define $d(x,L)$. To satisfy condition (I) $d(x,L)$ must satisfy the property

$$d(x,L) = 0, \text{ if } x \in L. \qquad --(4)$$

From (3) and (4) it follows that

$$D(L_1,L_2) = \sum_{x \in L_1 - L_2} p(x)\ d(x,L_2) + \sum_{y \in L_2 - L_1} q(y)\ d(y,L_1).$$

In general, we will define $d(x,L)$ in the following form,

$$d(x,L) = \sum_{y \in L} f(x,y,WLD(x,y)),$$

where $WLD(x,y)$ is the Weighted Levenshtein Distance between the two strings x and y. An example is

$$d(x,L) = \sum_{y \in L} a(x,y)\ g(WLD(x,y)). \qquad --(5)$$

Here $a(x,y)$ is a weight factor, called the window function. As L can be an infinite language, the weight factor limits the sum to only a finite number of terms. We make $a(x,y)$ depend on the length of string x, denoted by $\ell(x)$, to bound the magnitude of $d(x,L)$. Therefore, there exists a constant c, for which

$$0 <= d(x,L) <= c, \ \forall x \in L_1.$$

Thus for any arbitrary language, L,

$$p(x)\ d(x,L) <= p(x)\ c <= c \qquad --(6)$$

$$p(x)\ d(x,L) >= 0 \qquad --(7)$$

Equations (6) and (7) show that (3) is bounded and therefore, convergent.

Equation (2) reduces to

$$D(L(G),S) = \sum_{L(G)-S} p(x)\ d(x,S) \qquad --(8)$$

where $d(x,S) = 0$, if $x \in S$. The above is usually an infinite sum, but because of its bounded and convergence properties, only a finite number of terms have to be computed to produce $D(L(G),S)$ for a specified accuracy.

4.3.2 Particular Proximity Measures

In this section we develop particular proximity measures defined in (2) that reflect the difference in structure between a string x and the set of strings in the language L. Three different discrepancy measures are proposed below.

The minimum discrepancy measure is defined as,

$$D_{min}(L_1,L_2) = \sum_{x \in L_1} p(x)\ d_{min}(x,L_2) + \sum_{y \in L_2} q(y)\ d_{min}(y,L_1).$$

where

$$d_{min}(x,L) = a(x)\ \min_{y \in L}\{WLD(x,y)\} \qquad --(9)$$

and $a(x) = 1 / \ell(x)$. Since

$$WLD(x,y) <= \ell(x) + \ell(y)$$

we have

$$\min_{y \in L}\{WLD(x,y)\} <= \ell(x) + b$$

where $b <= \min\{\ell(y)\}$, which shows that d_{min} is bounded, which implies that $D_{min}(L_1,L_2)$ is convergent.

The minimum discrepancy measure can be looked upon as an optimistic estimate of the proximity between two languages. Another estimate may be an average distance between string, x, and the set, L. A reasonable way to define the distance would be to compute a weighted average distance between x and all strings from a finite subset of L. Such discrepancy measures are called window discrepancy measures. Finite subsets are identified by placing windows on the set L which assigns weights to all strings whose lengths are between $\ell(x)-c$ and $\ell(x)+c$, where $c >= 0$. The general definition of window discrepancy measures $d_*(x,L)$, appears below. Let

$$L_c(x) = \{y \in L \mid \ell(x)-c <= \ell(y) <= \ell(x)+c\}, \text{ where } c >= 0$$

If $x \in L$, $d_*(x,L) = 0$.

If $x \notin L$ and $L_c(x) = \phi$, define $d_*(x,L) = 2 + c/\ell(x)$.

If $x \in L$ and $L_c(x) \neq \phi$, $d_*(x,L) = \sum_{y \in L} a_*(x,y)\ WLD(x,y)$,

where $a_*(x,y) = 0$, if $y \notin L_c(x)$.

Let $K_c(x)$ denote the cardinality of $L_c(x)$. For some fixed $c >= 0$, the rectangular window function is,

$$a_{rect}(x,y) = \begin{cases} 1 / (\ell(x)\ K_c(x)), & \text{if } y \in L_c(x) \\ 0 & \text{, otherwise.} \end{cases}$$

The corresponding discrepancy is

$$d_{rect}(x,L) = \begin{cases} 0, & \text{if } x \in L \\ 2 + c/\ \ell(x), & \text{if } L_c(x) = \phi,\ x \notin L \\ [1/\ \ell(x)]\ \overline{WLD(x,c)} & \text{otherwise,} \end{cases} \qquad --(10)$$

where $\overline{WLD(x,c)}$ denotes the average distance between x and strings in $L_c(x)$. Similarly a triangular window function is defined as

$$a_{tri}(x,y) = 1 \Big/ (\ell(x).K_c(x)) \begin{cases} (\ell(y) - \ell(x)+c)/c; & \ell(y) <= \ell(x),\ y \in L_c(x) \\ (\ell(x) - \ell(y)+c)/c; & \ell(y) > \ell(x),\ y \in L_c(x) \\ 0 & , y \notin L_c(x) \end{cases}$$

Strings in $L_c(x)$ with lengths close to $\ell(x)$ influence the triangular proximity most. The corresponding discrepancy function is defined as

$$d_{tri}(x,L) = \begin{cases} 0, & \text{if } x \in L \\ 2 + c/\ell(x), & \text{if } L_c(x) = \phi,\ x \notin L \\ a_{tri}(x,y)\ WLD(x,y), & \text{if } x \notin L. \end{cases} \qquad --(11)$$

From a comparison of the three discrepancy measures it is apparent that

$$d_{min}(x,L) <= d_{rect}(x,L);$$

$$d_{min}(x,L) <= d_{tri}(x,L),$$

Therefore it follows that

$$D_{min}(L_1,L_2) <= D_{rect}(L_1,L_2);$$

$$D_{min}(L_1,L_2) <= D_{tri}(L_1,L_2).$$

For a fixed window size, $c >= 1$, we have

$$D_{tri}(L_1,L_2) <= D_{rect}(L_1,L_2).$$

This confirms the fact that the minimum discrepancy measure is an optimistic estimate of the proximity. The triangular and rectangular measures fall into the category of weighted average estimates. The next section shows that all three discrepancy measures behave similarly when used with an inference scheme. The choice of a particular measure depends on the application being considered.

4.3.3 Properties of Structural Discrepancy Measure

We now look into properties of a structural discrepancy measure $D(L(G),S)$ in the grammatical inference problem. The greater the probability or the smaller the length of a string, the larger is its contribution to the discrepancy measure. This conforms to intuitive notions [42] that small strings are more important than large ones. Simple examples are used to illustrate some of the desired properties. We assume

$$S \subset L(G) \text{ and } D(L(G),S) = \sum_{x \in \{L(G)-S\}} p(x)\ d(x,S).$$

Property: Given a probabilistic language, L, and two finite sample sets S_1 and S_2, such that

$$S_1 \subset S_2 \subset L. \text{ Then,}$$

$$D_{min}(L,S_1) >= D_{min}(L,S_2).$$

The proof is simple.

The same conclusion cannot be made for D_{rect} (and D_{tri}) as demonstrated below.

Example 1: Let L = {abc, bac, cba, acb, bca, cab}, with probabilities

(1/5.1, 1/51, 1/5.1, 1/5.1, 1/5.1, 1/5.1),

S_1= {abc, cba} with probabilities, (1/2, 1/2) and

S_2= {abc, bac, cba} with probabilities, (1/3, 1/3, 1/3).

With window size = 0, $D_{rect}(L,S_1) = 1.2156$ and $D_{rect}(L,S_2) = 1.3072$.

Therefore, $D_{rect}(L,S_2) > D_{rect}(L, S_1)$ though $S_1 \subset S_2$.

For a language, L, and and a finite sample set $S \subset L$, we can write our three different discrepancy measures as,

$$D_{min}(L,S) = \sum p(x) \quad \frac{1}{\ell(x)} \quad \min_{y \in S}\{WLD(x,y)\} \qquad --(12)$$

$$D_{rect}(L,S) = \sum p(x) \quad \frac{1}{\ell(x)} \quad \overline{WLD(x,c)} \qquad --(13)$$

$$D_{tri}(L,S) = \sum p(x) \quad \frac{1}{\ell(x)} \quad \overline{\overline{WLD(x,c)}} \qquad --(14)$$

where $\overline{\overline{WLD(x,c)}} = \sum_{y \in L} a_{tri}(x,y)\ WLD(x,y)$

The contribution of a string $x \in L$ to the discrepancy measure is a product of three factors - its probability, the inverse of its length and its proximity measure to the set, S. This shows that the

discrepancy measures increase in direct proportion to the probability and the WLD function of the strings in L(G)-S, and shorter strings have a greater contribution to the discrepancy measure. An example below illustrates the convergence property of the discrepancy measure for infinite languages.

Example 2: Consider $S = \{a, a^2, -----, a^n\}$ with

probabilities (1/n, 1/n, ----, 1/n).

Let G: X -> aX | a (p,1-p),

so $L(G) = \{a^n \mid n >= 1\}$ with probability$(a^n \epsilon L(G)) = p^{n-1}(1-p)$.

Then, $D_{min}(L(G),S) = \sum_{j=n+1}^{\infty} (1-p)\,p^{j-1}(j-n)/j =$

$$1/p - (1-p^n) - n(1-p)/p[\log(1/(1-p)) - \sum_{j=1}^{n}(p^j/)\,j]$$

This shows that as n increases, $D_{min}(L(G),S)$ decreases. As $n \to \infty$, $D_{min}(L(G),S) \to 0$.

4.3.4 Computation of the Discrepancy Measure

In this section we discuss the computational complexity of the discrepancy measure in terms of the number of steps required in the computation. This discussion pertains to sequential algorithms for computing the WLD. Recently [17] parallel algorithms have been proposed that reduce the order of the time complexity. They are discussed in detail in Chapter 6.

The sum in $D(L(G),S)$ usually involves an infinite number of terms, because $L(G)$ is generally an infinite language. Following Liou [66], there are two ways in which we can make the computation finite. One way is to use the fact that any grammar generates only a finite number of strings of length <= m, where m is a positive integer. By choosing a subset,

$$A_m = \{x \in L(G) \mid \ell(x) <= m\}, \text{ we can approximate}$$

$$D(L(G),S) \approx D(A_m,S) = \sum_{x \in A_m} p(x)\ d(x,S)$$

One intuitive choice for m is $\max\{\ell(y)\}$, but then the computed discrepancy measure may fail to verify whether $L(G)$ is a natural generalization of S.

A second approach is to use string probabilities to limit the number of strings used in the computation. Select a real number, q, $0<q<1$, and demand that $(100q)\%$ of the probability of strings from $L(G)$ be used in the discrepancy computation. This requires choosing strings from $L(G)$ in a prescribed order, until the sum of the probabilities equals or just exceeds q. A reasonable way to do this is to generate strings from $L(G)$ in lexicographic order and accumulate these strings in a set B_q, till $p(x) >= q$. Then the discrepancy measure can be approximated as

$$D(L(G),S) \quad D(B_q,S) = \sum_{x \in B_q} p(x)\ d(x,S).$$

Distance computations are based on the WLD [80]. The conventional WLD algorithm has computational complexity of order nm, where n and m are string lengths. Masek et al. [71] have proposed a faster algorithm of order (n max(1,n/log m)), where n >= m. If the sample set, S, has M strings and we restrict ourselves to N strings from L(G), the worst case computation time for $D_{min}(L(G),S)$ becomes O(MNmn) (or O(NMn max(1,n/log m)), n>=m). For rectangular and triangular windows the order of computations is similar, except that for x ∈ S, the actual computation time may be less than required for $D_{min}(L(G),S)$.

The computation time for $D_{min}(L(G),S)$ can be reduced by use of a search algorithm. The algorithm is summarized as follows -

Arrange the strings in S in order of their lengths.

For each x ∈ L(G) chosen in the computation first compare x with

$y \in S : \ell(y) = \ell(x)$.

If $\nexists\, y : WLD(x,y) = 0$, exit and pick the next x.

Otherwise, find a number a : $a = \min_{y \in S \text{ such that } \ell(x)=\ell(y)} \{WLD(x,y)\}$.

The search can then be restricted to the set

$$\{ y \in S : \ell(x)-a <= \ell(y) <= \ell(x)+a \}.$$

For every new length of strings being searched, the bound a is updated, if $a_{new} < a_{old}$, and correspondingly the search space is minimized.

All the discrepancy measures defined above are of the same order of complexity; the computation time for rectangular and triangular window measures depends on the window size. All three measures have identical properties in relation to a string grammar inference scheme. We chose the minimum discrepancy measure as our proximity measure for structural discrepancies, but test runs for the experiments we conducted in Chapter 6 showed that the other two measures would have produced almost identical results.

4.4 Discrepancy Measure for Array Languages

The fifth step in our inference scheme described in Section 3.2 requires the establishment of a measure of suitability to determine how well the language of the inferred array grammar fits the training set of patterns. We now extend the concept of the structural discrepancy measure for string languages to a structural difference measure between two array languages. This requires the definition of a proximity measure between two arrays. The string edit or the WLD distances has been extended to measure the distance between two finite arrays of symbols [76,109] and is called the two dimensional WLD, or WLD2. We use the concept of WLD2 to define the discrepancy measure between two array languages. The computational aspects of this measure are investigated, and then approximations to the actual measure are

developed to reduce the computational effort in our inference scheme.

The discrepancy measure between an inferred array language L and the training set of array patterns S is defined as

$$D(L,S) = \sum_{x \in L-S} p(x)\ d(x,S) \qquad --(15)$$

where x is an array pattern and p(x) is the probability that $x \in L$; d(x,S) is a proximity measure that has the same properties as the proximity measures of Section 4.3.1 and

$$d(x,S) = \sum_{y \in S} a(x,y)\ g(WLD2(x,y))$$

where a(x,y) is the weight factor, or the window function. As in Section 4.3.2 we can define the minimum measure and the two window measures.

The language L, as a generalization of S, is often an infinite language, so we need to adopt techniques to make the computation finite. The algorithm to compute WLD2 between two arrays of sizes IxJ and KxL, is of order $(I^2+JK^2L+IJ^2KL^2)$ [109]; if I = J = K = L = N, we have $O(N^6)$ and this makes the computation formidable even for moderate N. We approximate D(L,S) by computing only a few terms from L-S. Suppose the training set S, contains array patterns of types N_1, N_1+1, ---, N_2 (the definition of type is introduced in Section 2.4), with n_{N_1}, n_{N_1+1}, ----, n_{N_2} array patterns of each type respectively. We define a new set $L_S \subset L$, such that L_S contains $2n_{N_1}$, $2n_{N_1+1}$, ---, $2n_{N_2}$ patterns of types N_1, N_1+1, ---, up to N_2, respectively. For each

size N_k, L_S contains the $2n_{N_k}$ patterns of highest probability from L; $D(L_S,S)$ is then used as an approximation for $D(L,S)$. For our inference scheme, we chose the triangular window proximity measure to compute $d(x,S)$, $x \in L_S$; the window is defined as

$$L_c(x) = \{y \in S \mid size(x)-c <= size(y) <= size(x)+c\}, \text{ with } c=1.$$

where size(x) denotes the size of array x. This estimate is a rough approximation to the actual discrepancy measure D(L,S), but it does estimate the fit for comparison of array grammars inferred by applying different noise and distortion models to the same training set without making the computations unduly lengthy. These experiments are explained in detail in Chapter 6. Much better approximations can be obtained, however, by the use of parallel systems and parallel algorithms, which greatly reduce the time complexity of the algorithm. They too are discussed in Chapter 6.

4.5 Summary

This chapter discusses the role of discrepancy and complexity measures in the inference of string grammars. A number of the existing measures are reviewed. A complexity measure is adopted that satisfies Blum's and Wharton's criteria, is computationally very simple, and quite adequate for regular grammars. A new structural discrepancy measure is defined; the total discrepancy between the language of an inferred grammar and the sample set is defined as a linear combination of the structural discrepancy measure between the two sets and the

absolute discrepancy measure between strings common to the two sets. Lastly, a new discrepancy criterion for the fit between two array languages is proposed. Computational aspects of these algorithms are investigated.

CHAPTER V

MODELS FOR NOISE AND DISTORTION

The inference of probabilistic grammars involves estimating probabilities for production rules from probabilities assigned to the samples in the training set. In the string grammar inference problem, probabilities are assigned to strings either by frequency counts [11,42], or from a probabilistic model. A probability model on strings defines conditional probabilities of the type $p(x|y)$, where $x,y \in T^+$. An example is the error correcting parsing model described in Section 3.1, where probabilities assigned to each of the three edit operations can be used to compute the probabilities of strings.

Pattern description and learning based on array grammars involves inferring the structure of a given class of patterns from images that may be corrupted by some physical phenomenon. The ideal image is changed by random fluctuations in the digitizing unit, or during transmission. This phenomenon, which is independent of the underlying pattern structure, is called a noise process. In other situations, the corrupting phenomenon may depend on the underlying pattern structure. Such effects are observed during photographing or digitizing objects when the intensity differences between light and dark areas is low, causing diffraction effects. This effect becomes more pronounced when

the aperture of the camera lens is small, resulting in blurred edges separating the object from the background [88]. Points near the boundaries of objects experience more distortion than internal regions. We term such processes distortion processes.

This chapter investigates some noise and distortion models. We begin with a simple, independent noise model, and then extend it to define two intuitive distortion models and a comprehensive distortion model using Markov random field theory. We study the mathematical formulation of each model, and techniques for generating sample patterns, based on an ideal structure along with the estimation of the parameters of a model. The estimated parameters are used to compute individual block probabilities as outlined in Section 3.2.

5.1 The Independence Noise Model

The independent noise model assumes that the probability of a pixel changing value is independent of all other pixel values in the pattern. This is equivalent to considering an ideal image, which is a digitized mxn pattern, being degraded with independent, additive, stationary noise.

The noise process is modelled as a Bernoulli process. The number of changes in a block is assumed to have a binomial distribution, with parameters b, the number of pixels in the block, and p, the probability

that a pixel value is switched.

The scheme for generating patterns from this noise process is described in Procedure 1 below. We assume that all pattern types (random variable N) occur with equal probability, so N has a discrete uniform distribution over $[N_1,N_2]$.

$$P[N=n] = \begin{cases} \dfrac{1}{N_2-N_1+1} & , \text{ if } N_1 <= n <= N_2 \\ 0 & , \text{ otherwise.} \end{cases}$$

Procedure 1:

Input: G, the rewriting rules for the ideal structure, p, N_1, N_2.

Begin:

(1) Generate $n \approx U[N_1,N_2]$

(2) Using the nonterminal rewriting rules, create an ideal pattern structure of size n.

(3) For each pixel in pattern,

(a) generate $x \approx U[0,1]$

(b) if x<= p, flip pixel value

otherwise, retain pixel value

End loop.

End

The result of applying this procedure to a set of isosceles triangles and the numeral '7' are illustrated in Figures 7 and 8 respectively. The parameter p was set at 0.05.

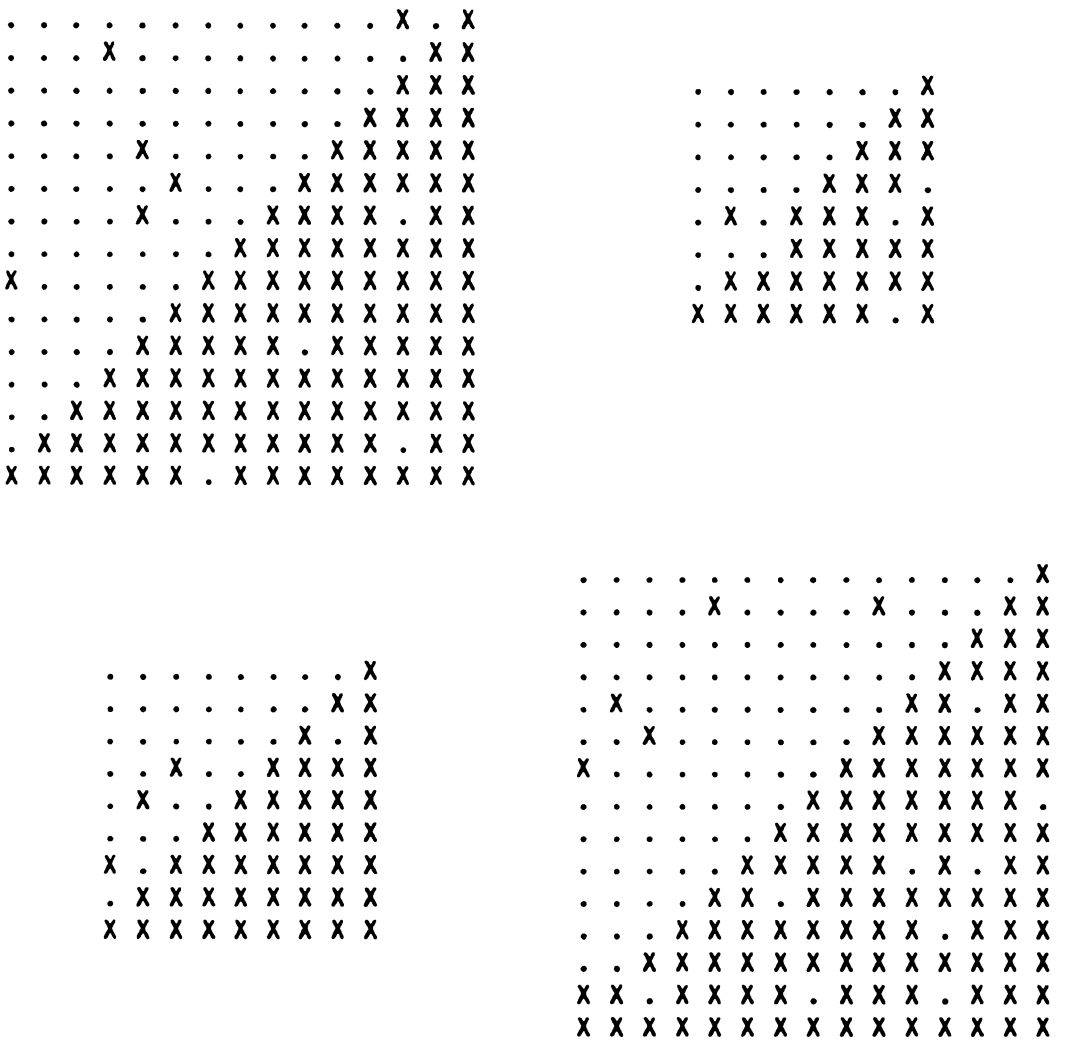


Fig. 7: Triangles Generated from Independent Noise Model. p = 0.05.

Given that the patterns in the training set are obtained by imposing the noise model on the ideal pattern structure, the maximum likelihood estimate of the parameter p, is

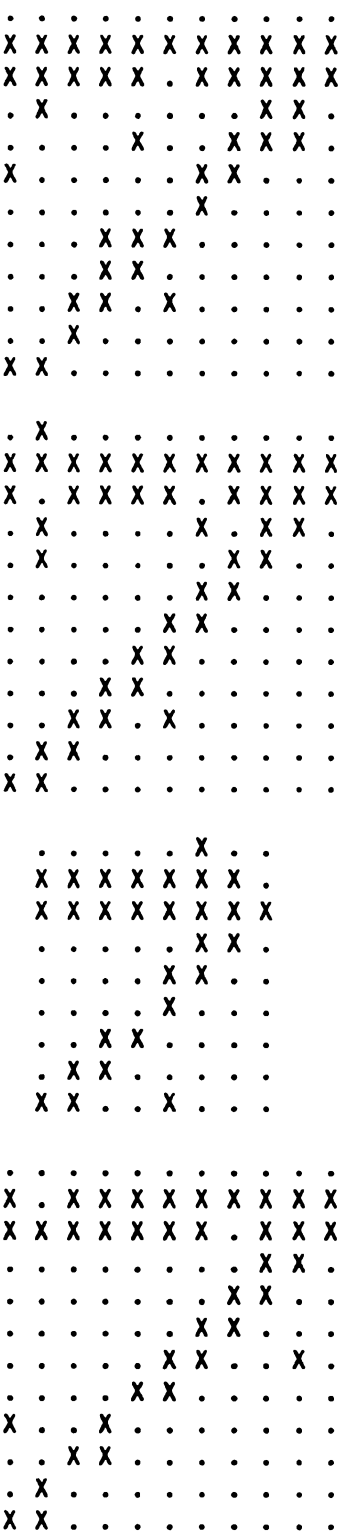


Fig. 8: Noisy 7's. p = 0.05.

$$p = \frac{\text{total number of pixels that are flipped}}{\text{total number of pixels in the sample set}}$$

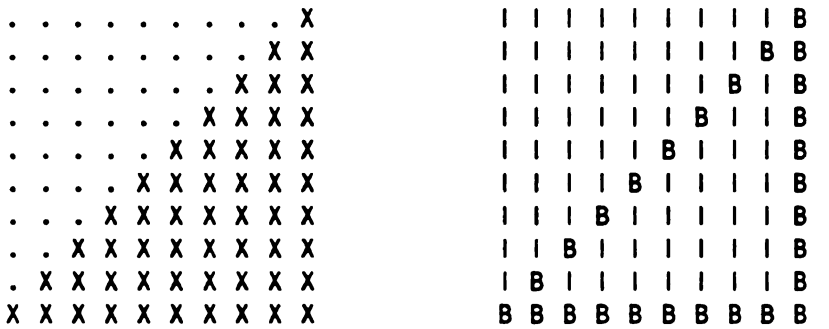
The ideal structure must be known. Our inference scheme requires that the patterns be broken down into blocks, which can then be grouped into sample patterns for the different intermediate languages. Treating each intermediate language separately, we can compute the probability of occurrence of each block from the noise model. The probability of observing a block with b pixels of which c are flipped is:

$$p^{c} (1-p)^{b-c} \qquad --(1)$$

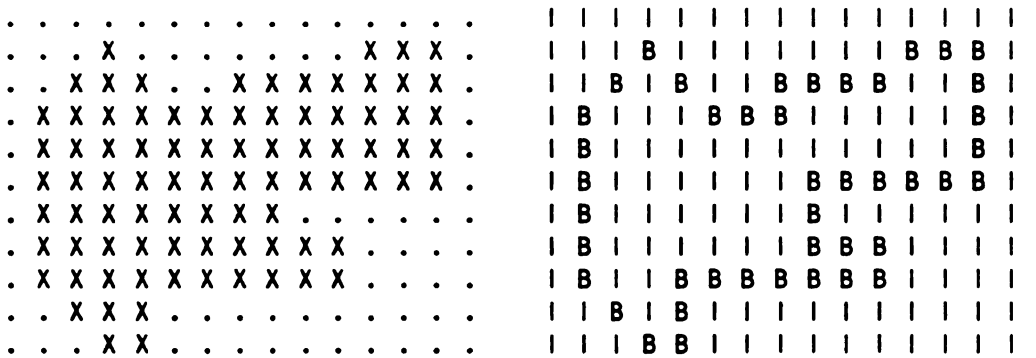
This model for noise is computationally very simple, both in terms of generating patterns, and computing the block probabilities required by the inference scheme. It represents random fluctuations in a digitizing unit or a transmission channel that are independent of the underlying pattern structure.

5.2 Intuitive Distortion Models

The corrupting process in distortion models depends on the structure of the ideal image. Deviations from the ideal are seen more along the boundary than in the interior of the object or in the background. Boundary and interior points are illustrated in Fig. 9. This type of corruption can be attributed to diffraction effects that occur when digitizing an object or by image preprocessing and segmentation operations that extract a binary image from a background



(a) Triangle



(b) Arbitrary Figure

Fig. 9: Interior and Boundary Points

of irrelevant detail. Two simple intuitive distortion models that are extensions of the independent noise model are presented in this section. We discuss their shortcomings and motivate the study of the distortion models based on Markov random fields.

5.2.1 Independence Distortion Model

Instead of looking upon an image or pattern as a homogeneous region, with the same probability of change at each pixel, we define a nonhomogeneous model with two parameters; p_i is the probability of change for interior points, and p_b is the probability of change for boundary points. If $p_b > p_i$ boundary changes are more likely than changes in the interior and background regions.

The scheme to generate distorted patterns using this model (Procedure 2) is similar to that for the independent noise model. The pattern type is assumed to have a $U[N_1,N_2]$ distribution.

Procedure 2:

Input: G, B, p_i, p_b, N_1 and N_2

Begin:

(1) Generate $n \approx U[N_1,N_2]$

(2) From rewriting rules create ideal structure of size n

(3) For each pixel in pattern,

(a) Generate $x \approx U[0,1]$

(b) if interior pixel then

if $x <= p_i$ flip pixel value

otherwise retain

otherwise (boundary point)

if $x <= p_b$ flip pixel value

otherwise retain

End loop.

End

An example in Figure 10 illustrates this procedure with the class of triangular patterns. The parameters p_i and p_b were set to 0.05 and 0.15, respectively.

Estimation of the parameters p_i and p_b, from a finite set of training patterns using maximum likelihood estimates, again involves counting procedures,

$$\hat{p}_i = \frac{\text{total number of interior pixels flipped}}{\text{total number of interior pixels}}$$

$$\text{and } \hat{p}_b = \frac{\text{total number of boundary pixels flipped}}{\text{total number of boundary pixels}}$$

Once $\hat{p}_i$ and $\hat{p}_b$ are estimated, a probability can be assigned to any block by assuming the patterns are acted upon by two independent binomial processes. The probability that a block b pixels experiences c_i internal changes and c_b boundary changes is

$$p_i^{c} (1-p_i)^{b-c} \; p_b^{c} (1-p_b)^{b-c}$$

where b_i is the total number of internal points in the block, and b_b is the total number of boundary points in the block; $b_i+b_b = b$.

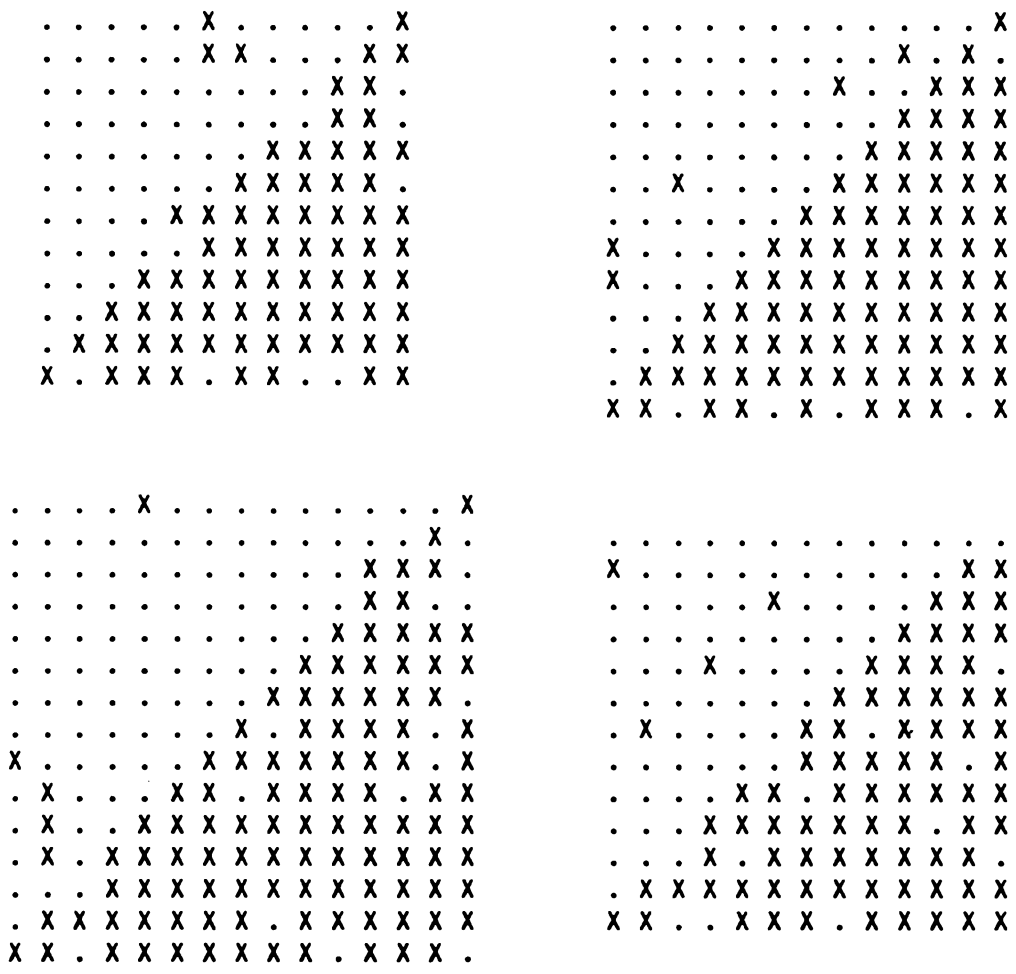


Fig. 10: Triangles generated by Independent Distortion Process. p_i=0.05, p_b=0.15

5.2.2 Neighborhood Distortion Models

The neighborhood distortion model allows the distortion process at a pixel to be a function of the values at neighboring pixels in the ideal pattern. The surrounding pixels determine the extent of

nonhomogeneity in that region. If a majority of the neighboring pixels have the same value, we assign a smaller probability of change than when the surrounding neighborhood is nonhomogeneous.

We now formally define the concept of neighborhood for a two dimensional array of pixels. The distance between pixels or points in the array assumes discrete values. Figure 11(a) depicts some of the squared distances from point x in the array. These squared distances form a sequence of numbers 1, 2, 4, 5, --- called $\{e(k)\}$, $k = 1, 2,$ ----. Following usual image processing conventions, pixel (m,n) is a k^{th} order neighbor of pixel (i,j) iff the squared distance between (m,n) and (i,j) is the k^{th} term in $\{e(k)\}$. Figure 11(b) shows the ordering of the neighboring pixels from the one marked x.

A neighborhood distortion model of order k requires that the probability of change for each pixel (i,j) depends on all its neighbors up to order k in the ideal pattern. In general, if the number of pixel neighbors of order k or less is c, we define a discrete neighborhood function W(i,j) for each pixel (i,j) that can assume at most 2^c different values. We assume, for a k^{th} order neighborhood function, that the pattern is surrounded by k rows and columns of 0's at each edge, so that W(i,j) is then well defined for all pixels of the pattern. Corresponding to this neighborhood function we have at most 2^c different probability of change values, $(p_1, p_2, ---, p_2)$, as parameters for the model. For example, consider a first order model

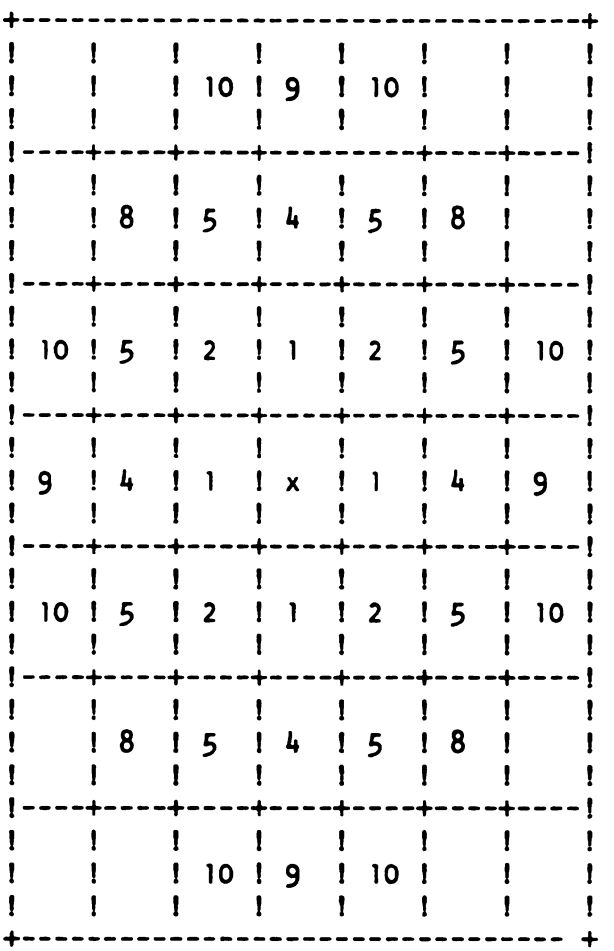


FIG. 11(a): Squared Distances to Point x.

where each pixel has four neighbors. The neighborhood function can have a maximum of 16 different values, and the model can have a maximum of 16 different probability of change values as parameters. Simpler functions can be used to describe the type of distortion we desire, such as,

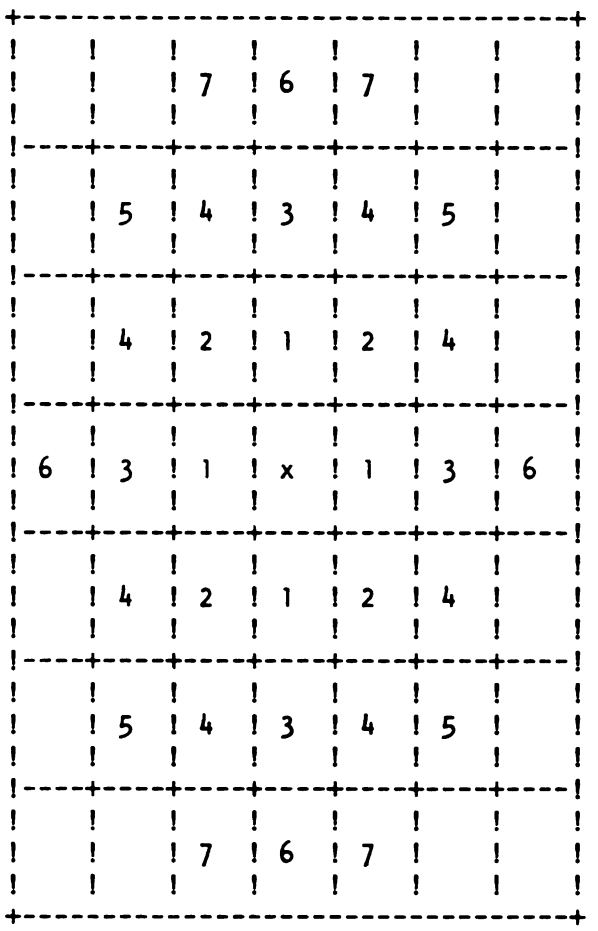


FIG. 11(b): Orders of Neighbors Relative to Point x.

$$W(i,j) = \sum_{(m,n) \in Nb_k(i,j)} v(m,n) \qquad \text{---(2)}$$

where $Nb_k(i,j)$ denotes pixels in the k^{th} order neighborhood of (i,j) and $v(m,n)$ is the value of pixel (m,n).

For k = 1, the above function can have five values, 0 through 4, so we can define a maximum of five different probabilities of change, p_0 through p_4. As a simple example, we set $p_0 = p_4$ and $p_1 = p_2 = p_3$. Then a value of 0 or 4 for the neighborhood function indicates a homogeneous region and the other values represent nonhomogeneous regions. Setting $p_1 > p_0$ produces more likely distortion along edges than in interior regions.

The value of the neighborhood function for each pixel is determined entirely from the ideal structure of the pattern, which is predetermined. Therefore, the probability of change for any pixel (i,j) is independent of the probability of change for any other pixel (i',j') in the same pattern. We define an indicator function z(i,j) for an observed pattern, as

$$z(i,j) = \begin{cases} 1 & \text{if the pixel value is flipped,} \\ 0 & \text{otherwise.} \end{cases}$$

Assuming that N, the type of the pattern, comes from a $U[N_1,N_2]$ distribution, the probability of a block sample belonging to an intermediate language is

$$\frac{1}{N_2-N_1+1} \sum_{(i,j)\in block} [z(i,j)p_{W(i,j)} + (1-z(i,j))(1-p_{W(i,j)})]$$

This model is uniquely defined by the neighborhood function W(i,j), the

size of the neighborhood k, and the different probability of change values that are used. The generation procedure for samples from this model is given in Procedure 3.

Procedure 3:

Input: G, W(i,j), $\{p_i\}$, N_1, N_2

Begin:

(1) Generate $n \approx U[N_1, N_2]$

(2) Using rewriting rules create ideal structure of type n

(3) For each pixel (i,j)

(a) compute W(i,j) from ideal pattern

(b) generate $x \approx U[0,1]$

(c) if $x <= p_{W(i,j)}$, flip pixel value in observed pattern

otherwise, retain ideal pixel value

End loop.

End

Maximum likelihood estimation of the parameters, given the neighborhood function and size of neighborhood, again involves a counting procedure. The pixels in the observed patterns are grouped in terms of their neighborhood function values, and for each distinct functional value, we count the number of pixels flipped as well as the total number of pixels in that group. Therefore, for a particular value of W(i,j) = x,

$$\hat{p}_x = \frac{\text{number of x-group pixels flipped}}{\text{total number of x-group pixels}}$$

The complexity of this model depends on the size of the neighborhood and the neighborhood function W. We chose the W function as defined in (2), and used the model for first order neighbors with two different probability of change values. The computational complexity is the same order as in the previous two models, except that we now compute the neighborhood function for each pixel (i,j). The results of applying this model to the triangular patterns are illustrated in Fig. 12.

5.2.3 Applications and Limitations

The two models described above satisfy the requirements for simple distortion processes in that the corrupting phenomenon depends on the underlying pattern structures. The neighborhood model is computationally more complex than the independence model. However, it is a richer model in the sense that it can model a variety of distortions. Both models are limited by the fact that they are static in nature. The pixel values in the observed patterns depend only on the ideal pattern structure, but are independent of each other and do not allow the propagation of distortion effects.

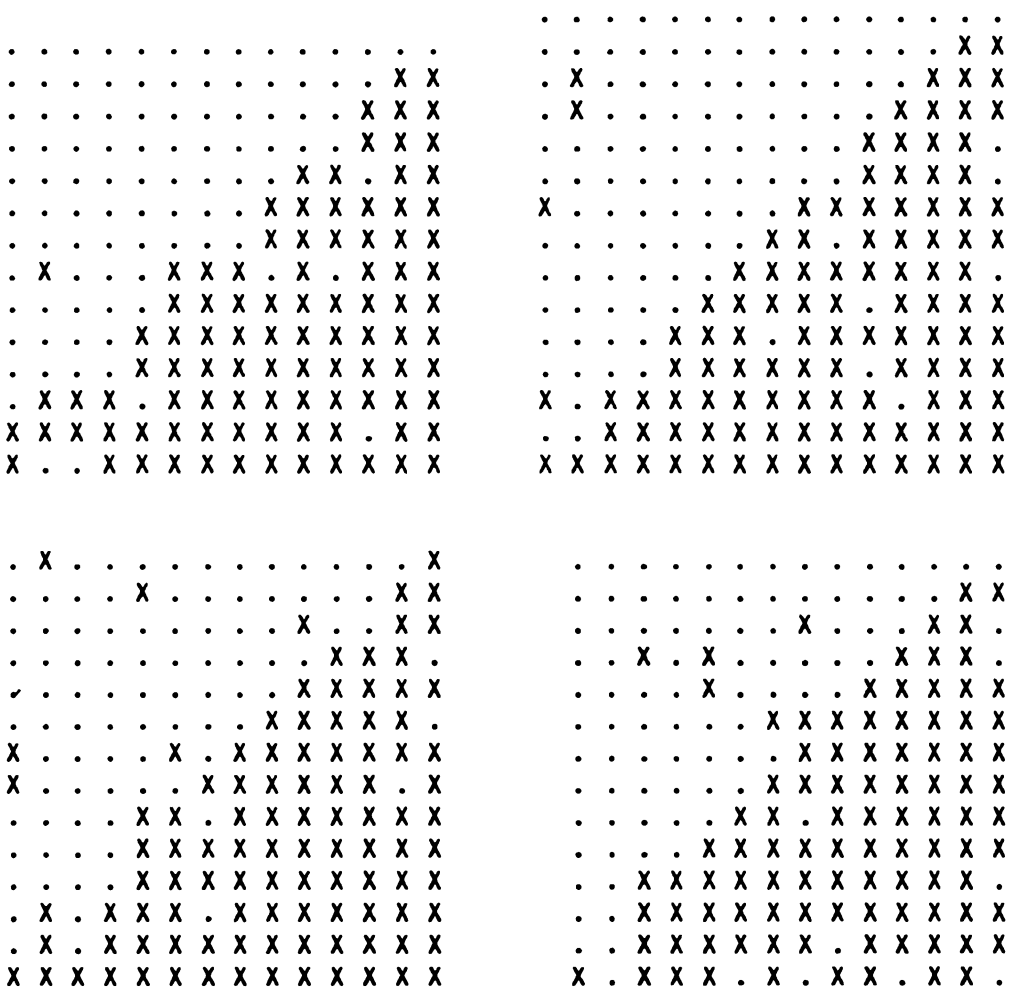


Fig. 12: Triangles generated by Neighborhood Distortion Process. $p_0 = p_4 = 0.05$; $p_1 = p_2 = p_3 = 0.15$

5.3 Markov Random Field Distortion Model

In this section we define a distortion model in which the probability that a pixel changes value depends on whether neighboring pixel values have also undergone a change. The mathematical formulation of this dependence is expressed as a Markov random field process [7,8,55]. The success of MRF's in modelling texture [25] suggests they are appropriate for distortion models.

Our two dimensional patterns are binary mxn arrays which, in Markov random field terminology are finite lattices whose sites are labelled 0 or 1. We begin this section with a brief review of the mathematical background of Markov random fields on finite lattices. Techniques for generating distorted samples using this model and the estimation of the parameters of this model from finite training sets are also studied.

5.3.1 Background

We follow the notations and definitions of Besag [8] and Isham [55]. The set of all possible realizations on a binary mxn lattice is denoted by T. A particular assignment of 0's and 1's to a lattice, called a configuration, is denoted $\underline{X} \in T$. Since T is finite, the process can be described by a discrete probability distribution, u, on the power set of T. Suppose $u(\underline{X}) > 0$, $\forall\ \underline{X} \in T$. Then, there exists a real valued potential function

$$Q(\underline{X}) = -\log\{ u(\underline{X}) / u(\phi) \} \qquad ---(3)$$

where ϕ is the configuration with 0's at all site values and $Q(\phi) = 0$. Rewriting (3) we obtain,

$$u(\underline{X}) = Z \exp\{-Q(\underline{X})\}, \qquad ---(4)$$

where Z is a normalizing constant obtained by setting the sum of (4) over all subsets $\underline{X} \in T$ to one. This type of probability distribution is frequently referred to as a Gibbs potential.

For finite lattices, the class of probability distributions with Gibbs potentials has been proved to be identical to Markov random field models [8,50,89]. A Markov random field [29] is a joint probability density on the set of finite mxn arrays, subject to the following conditions -

(i) Positivity : $p(\underline{X}) > 0 \quad \forall\, \underline{X} \in T$, and

(ii) Markovianity : $p(X_{i,j} \mid$ all points on lattice except $(i,j)) = p(X_{i,j} \mid$ neighbors of $(i,j))$.

A Markov random field is defined to be of order k, if $X_{i,j}$ depends on neighbors up to the k^{th} order.

The potential function can be expanded as follows if the pixels are labelled 1 to N.

$$Q(\underline{X}) = \sum_i x_i F_i(x_i) + \sum_{i,j} x_i x_j F_{i,j}(x_i,x_j) + \text{----}$$

$$+ x_1 x_2 \text{----} x_N F_{1,2,\text{----},N}(x_1,x_2,\text{----},x_N),$$

where the functions $\{F_{i,j,---,k}\}$ are called interacting potentials, and x_i is the label on the the i^{th} pixel.

A Markov random field is called an Auto-model if the following two conditions are satisfied.

(i) All F functions with more than two subscripts vanish identically;

(ii) The conditional probability distribution at each point belongs to the exponential family.

The potential function for a first order Auto-model is called the near neighbor potential function and can be expressed as

$$Q(\underline{X}) = \sum_i x_i a_i + \sum_{i,j} x_i x_j b_{i,j} \qquad ---(5)$$

where $b_{i,j} = 0$, unless x_i and x_j are neighbors. From (5) we obtain

$$p(x_i = x \mid \underline{X}^i) = \frac{\exp\{-x(a_i + \sum x_j b_{i,j})\}}{1 + \exp\{-(a_i + \sum x_j b_{i,j})\}} \qquad ---(6)$$

where $\underline{X}^i$ represents the configuration $\underline{X}$ minus the i^{th} point.

Equations (5) and (6) define a first order random field with N(N-1) parameters, the a_i's and the $b_{i,j}$'s. A Markov random field is said to be spatially homogeneous if

$a_i = a$, $\quad i = 1, 2, ---, N$;

$b_{i,j} = b(1)$ if x_i and x_j are vertical neighbors and

$b_{i,j} = b(2)$ if x_i and x_j are horizontal neighbors.

A spatially homogeneous first order Markov random field can be defined by these three parameters and is called isotropic if $b(1) = b(2)$.

5.3.2 The Distortion Model

We base our distortion model on the isotropic first order Auto model. Given an ideal structure of size mxn, we define a change template T as an mxn binary array generated by the isotropic first order Auto model that relates the ideal structure $Y = [y_{i,j}]$ and the observed patterns $X = [x_{i,j}]$ in the following manner,

$$x_{i,j} = \begin{cases} y_{i,j} & \text{if } t_{i,j} = 0 \\ 1-y_{i,j} & \text{if } t_{i,j} = 1. \end{cases} \qquad ---(7)$$

A desirable property for our distortion models is that the probability of change on and around borders be greater than the probability of change in interior and background regions. This is achieved by defining a nonhomogeneous Markov random field. The points on the change template are broken up into interior points and border points, as determined by the ideal pattern structure. Our first order nonhomogeneous Markov random field model requires five parameters,

(i) a_i - corresponding to first order interactions for the interior points,

(ii) a_b - corresponding to first order interactions for the

boundary points

(iii) $a_{i,i}$ - corresponding to pairwise interactions of interior points

(iv) $a_{b,b}$ - corresponding to pairwise interactions of boundary points, and

(v) $a_{b,i} = a_{i,b}$ - corresponding to second order interactions for interior-boundary pairs.

Setting $a_b < a_i$ produces more changes around border regions than in interior regions. Cross[25] showed that the parameters b(1) and b(2) control the clustering or grouping of 1's on a lattice for homogeneous models, with negative values indicating attraction and postive values indicating repulsion. Setting the parameters, b(1) and b(2) to 0, produces equally likely configurations. This interpretation is also valid for our nonhomogeneous model, and $a_{i,i}$, $a_{b,b}$ and $a_{b,i}$ control the clustering of interior-interior, boundary-boundary and interior-boundary pixels that are 1's.

Following (5) the potential function can be written as

$$Q_t(\underline{X}) = s_i a_i + s_b a_b + s_{ii} a_{i,i} + s_{bb} a_{b,b} + s_{bi} a_{b,i}, \qquad --(8)$$

where s_i and s_b are the numbers of interior and boundary points respectively that are 1's, and s_{ii}, s_{bb} and s_{bi} are the number of interior-interior, boundary-boundary and interior-boundary neighbor pairs, respectively, that are both 1's. The probability of occurrence of a block, obtained by applying the nonterminal rules of the pattern class, can be written as,

$$\frac{1}{N_2-N_1+1} \prod_{(i,j)\in block} p(t_{i,j} \mid \underline{X})$$

where $p(t_{i,j} \mid \underline{X})$ is defined in (6), and $t_{i,j}$ is the pixel value in change template T. We again assume that the type of a pattern is distributed as $U[N_1,N_2]$.

5.3.3) Generation of Distorted Patterns

The procedure for generating distorted patterns is based on an algorithm for generating texture [25]. The generation procedure is derived by analogy to a discrete, finite state Markov chain, whose states are the set of configurations $\{ \underline{X} \}$, with the limiting distribution given by $u(\underline{X})$ [49]. The transition probability from a state $\underline{X}$ to a state $\underline{Y}$ is given by $u(\underline{Y})/u(\underline{X})$. The algorithm is given in Fig. 13 [37,73]. Convergence to the limit distribution is unaffected by the choice of the initial configuration; only the time taken to achieve the final configurations depend on the initial configuration. Step 1 in Fig. 13, therefore, requires the choice of a good initial configuration. We generate our initial configuration using the procedure for the independent distortion model described in Sec. 5.2.1. This procedure requires the two probabilities p_b and p_i whose maximum likelihood estimates are developed below. A likelihood function that defines the ratio of the desired density function to that for the independent model distribution is,

$$L(\underline{X}) = \frac{\exp(s_i a_i + s_b a_b + s_{ii} a_{i,i} + s_{bb} a_{b,b} + s_{bi} a_{b,i}}{p_b^{s_b}(1-p_b)^{n_b - s_b} p_i^{s_i}(1-p_i)^{n_i - s_i}} \quad --(9)$$

where n_b and n_i are the total number of boundary and interior points respectively. Equation (9) can be broken up into product terms and expressed as

$$L(\underline{X}) = [\exp(-a_b)(1-p_b)/p_b]^{s_b} [\exp(-a_i)(1-p_i)/p_i]^{s_i}$$
$$[(1-p_i)^{-n_i} (1-p_b)^{-n_b} \exp\{-(s_{ii} a_{i,i} + s_{bb} a_{b,b} + s_{bi} a_{b,i})\}]$$

The two probabilities p_b and p_i are chosen to make the expressions in the first two brackets equal to 1. Therefore,

$$p_b = \frac{1}{1+\exp\{a_b\}}, \qquad p_i = \frac{1}{1+\exp\{a_i\}}$$

We follow the procedure used by Cross [25] to define the parameter STABLE. Given mxn arrays, we define M = mn, and consider M attempted switches to constitute one iteration. This count ignores attempted switches between pixels with the same value. Convergence to the desired configurations is indicated when the number of changes per iteration drop off to a small percentage of M, or the estimated parameters match the input parameters closely. These two factors are used to define the variable STABLE in Fig. 13. The algorithm was used to generate distorted sets for the triangle pattern structures as illustrated in Figs. 14 and 15 for different second order parameters. The time taken to generate templates of types 10-20 was roughly 500-600 seconds per template on a PDP-11/34 computer; the computation time for

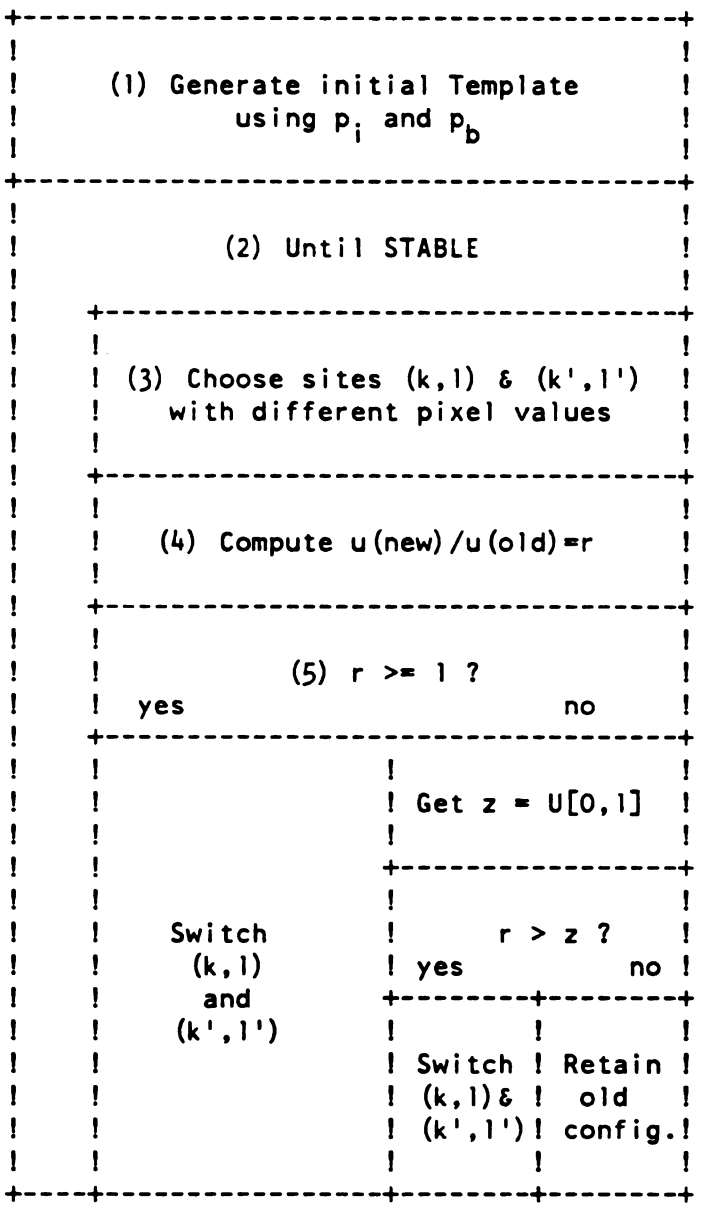


FIG. 13: Algorithm for Generating Template from MRF distribution with given Parameters

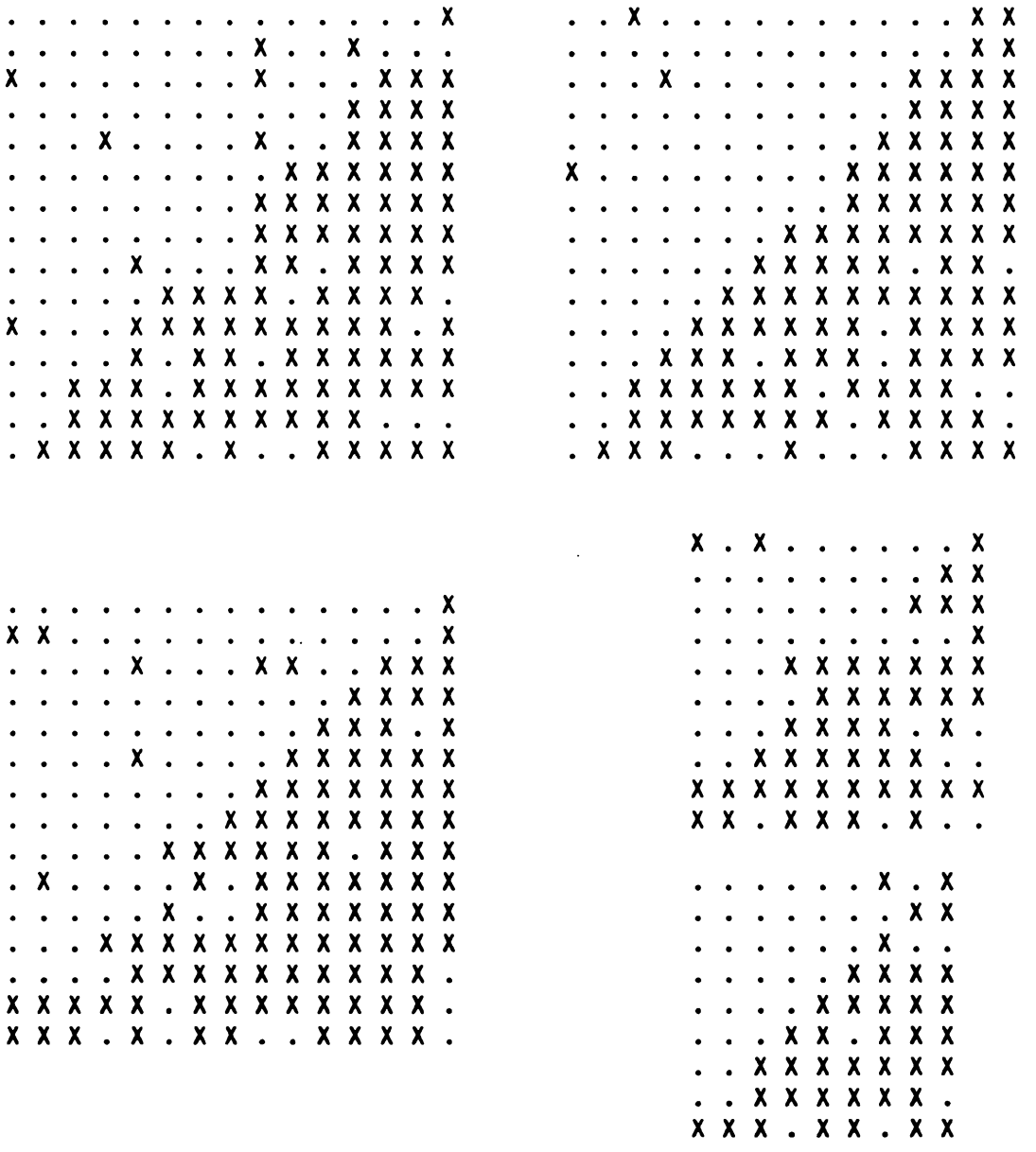


Fig. 14: Triangles created by Markov random field generative model. Parameters = < 2.2,1.5,0.1,-1.0,-0.8>.

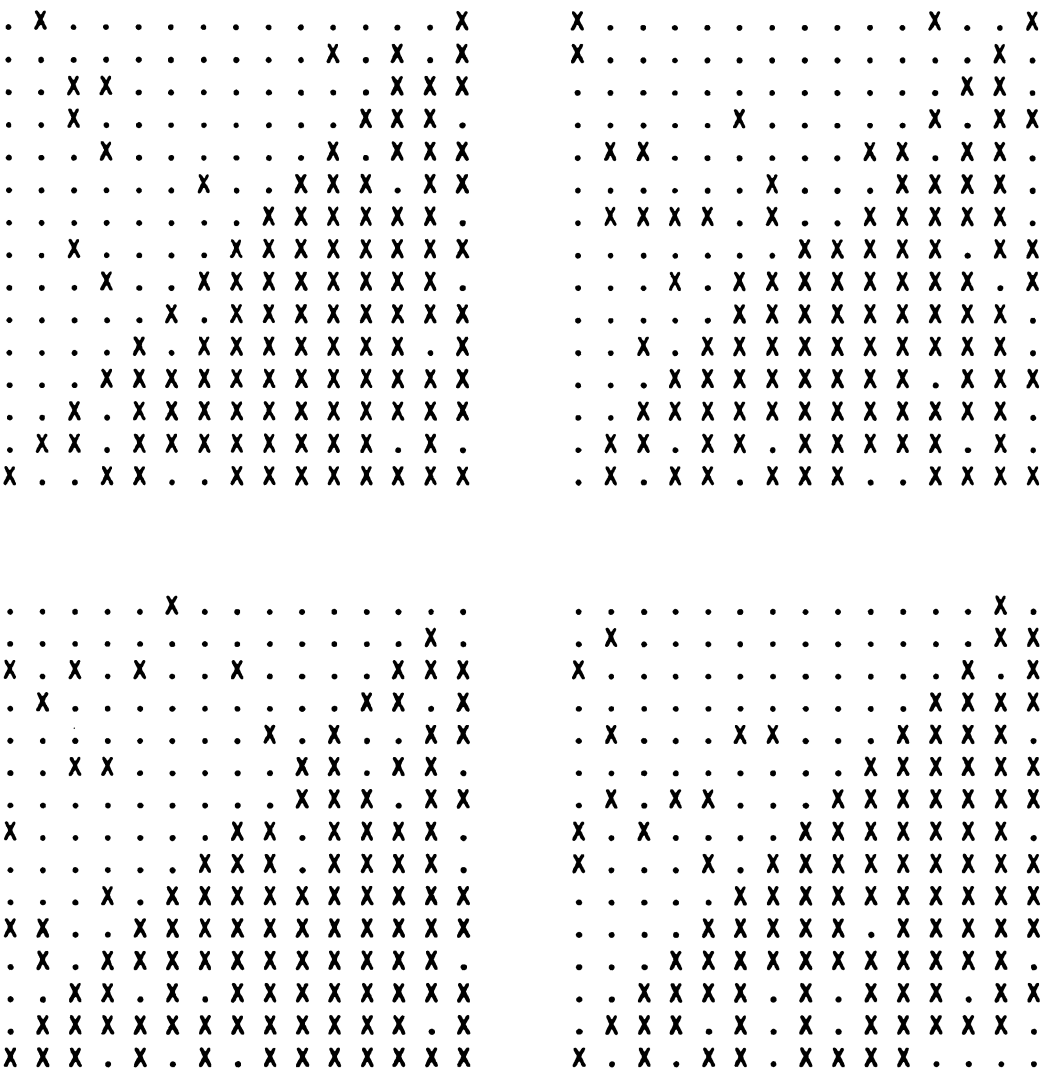


Fig. 15: Triangles from Markov random field generative Model. Parameters = <2.0,1.2,-0.5,-0.8,0.5>

the indepedent models was just a fraction of a second.

5.3.4 Estimation of Parameters

This section explains a method for estimating Markov random field parameters from a given set of training patterns. Nonhomogeneous Markov random fields involve many more parameters than homogeneous ones [25]. The first approach simplifies the computational requirements of the estimation procedure by approximating the probability distribution $u(\underline{X})$ of a configuration $\underline{X}$

$$u(\underline{X}) = \frac{\exp\{-Q_t(\underline{X})\}}{\sum_{\underline{X}\epsilon T}\exp\{-Q_t(\underline{X})\}} \qquad ---(10)$$

For convenience of notation we define C, the count vector as

$$C = (-s_i, -s_b, -s_{ii}, -s_{bb}, -s_{bi})$$

and A, the parameter vector as,

$$A = (a_i, a_b, a_{i,i}, a_{b,b}, a_{b,i})$$

Then $\exp\{-Q_t(\underline{X})\} = \exp\langle C,A\rangle$, where $\langle\ ,\ \rangle$ denotes the vector dot product. The denominator of (10) can be expressed as

$$\sum_{\underline{X}\epsilon T}\exp\{-Q_t(\underline{X})\} = 2^{mn}E[\exp\langle C,A\rangle] \qquad ---(11)$$

for a mxn lattice where E is the expectation assuming all

configurations generated by the parameter set A are equally likely.

We expect the count vector C to converge to a multivariate normal random vector as the pattern size increases. Monte Carlo runs performed to verify this for the triangular pattern structure showed that (11) seems to converge in distribution to a multivariate normal distribution with mean vector E[C] and covariance matrix R; R is a 5x5 matrix consisting of the second order moments of the C vector. Mardia's [26,70] skewness and kurtosis tests for multivariate normality were the criteria. The Monte Carlo runs involved picking a template size N, and then creating sets of 500 random templates at random. The count vector was then computed for each of these configurations, and Mardia's test was applied to each set of 500 count vectors. The procedure was repeated 100 times and the number of rejections of the null hypothesis for the skewness and kurtosis tests were observed. This test was conducted for pattern types 10, 15, 20, 40 and 50. For N less than 20, the multivariate normal hypothesis was consistently rejected by the skewness test but there were less than five rejections by the kurtosis test at the 5% significance level. For pattern types 40 and 50 the hypothesis was accepted at the 5% significance level for both the tests. Therefore, the approximation to the denominator seems adequate and we write,

$$E[\exp\langle C,A\rangle] \quad \exp\{\langle E[C],A\rangle + 0.5\langle A,RA\rangle\}$$

which reduces (10) to

$$u(\underline{X}) = \exp\{\langle C-E\langle C\rangle,A\rangle - 0.5\langle A,RA\rangle\}$$

If R is nonsingular, the maximum likelihood estimate can be written as

$$A = R^{-1}(C-E[C])$$

where C is the vector of observed values; E[C] and R depend on the ideal pattern structure.

Table 1 indicates that this estimation procedure worked poorly for the configuration of triangles. One possible explanation is that the parameters chosen generate distorted, atypical configurations and that the degree of nonhomogeneity is very high. There is a high degree of attraction among boundary 1's in the change template, but the distribution of 1's in the interior is quite random. The normal approximation of the probability function is exact at the origin of the parameter space. As one moves further away from the origin this approximation worsens. This was illustrated when the estimation procedure was applied to sets of triangular patterns of type 15 generated by the Markov field model with the parameters listed in Table 1. A second explanation is that the approximation is linear but the function is not, so the parameters are not unique; i.e., several parameter sets produce the same probability values but for different sets of configurations. The poor results obtained in Table 1 prompted us to use the exact form of the likelihood function to estimate the five parameters.

The log likelihood function for the probability distribution can be written as

Table 1: Parameter Estimates by the two schemes for the Markov random field model

	PARAMETER ESTIMATES - $< a_i, a_b, a_{i,i}, a_{b,b}, a_{b,i} >$		
	GENERATIVE	ESTIMATED	
		SCHEME 1	SCHEME 2
1)	0.0,0.0,0.0,0.0,0.0	0.0,-0.2,0.0,0.1,0.0	**
2)	1.0,1.0,0.0,0.0,0.0	2.5,2.5,-0.8,-1.1,-0.8	1.2,1.2,0.0,0.1,0.0
3)	2.0,1.0,0.0,0.0,0.0	5.4,3.6,-2.1,-1.6,-1.6	1.8,1.0,0.1,0.3,0.1
4)	2.0,1.0,-1.0,0.0	5.5,2.9,-2.1,-1.6,-1.6	1.8,1.3,0.0,-1.3,0.1
5)	2.0,1.2,0.0,0.0,-1.	5.4,3.8,-2.0,-1.6,-1.8	2.2,0.9,-0.1,0.2,-0.2
6)	2.2,1.5,0.1,-1.,-.8	6.4,4.4,-2.5,-2.3,-2.0	2.0,1.2,0.1,-1.1,0.7
7)	2.2,1.5,0.5,0.5,0.5	5.4,4.4,-2.1,-2.1,-1.7	1.5,1.3,0.5,0.7,0.5
8)	2.2,1.5,-.5,-.5,-.5	5.9,4.4,-2.3,-2.1,-1.9	1.6,1.5,-0.4,-0.4,-0.4
9)	2.2,1.5,-.2,-.8,.1	6.3,4.7,-2.3,-2.5,-1.9	1.7,1.4,-0.3,-0.9,-.05

** - could not compute; problems in numerical computation.

$$L = \sum_{x_i \in \underline{X}} \ln(p(x_i = x \mid \underline{X})$$

where $p(x_i = x \mid \underline{X})$ is given in equation (6). All the terms in the summation above are not independent, because of the nature of the Markov random field scheme. Therefore coding schemes [8,25], which break up the lattice into disjoint sets of independent points, are employed. A first order scheme requires at least two codings for

estimation purposes so we break up L into parts $L(1)$ and $L(2)$, corresponding to the two codings. The maximum likelihood estimate requires the solution of five simultaneous non linear equations for each k. This system of equations can be solved numerically by a multivariable extension to Newton's method [54]. The final estimate for A is the average value of the estimates obtained from each coding scheme. Although the estimation procedure is quite complex, it does yield accurate estimates for the parameter set A as illustrated in Table 1. The estimate may be more accurate as the pattern size increases.

5.4 Summary

In this chapter we have studied the independent noise model, two independent distortion models and a distortion model based on Markov random fields and presented their capabilities and restrictions. The Markov random field model is very rich, and generalizes other models. For example, setting the $a_{\cdots}$ parameters to 0 and $a_i = a_b$, creates the independent noise model; setting the $a_{\cdots}$ parameters 0, but requiring $a_i \neq a_b$ describes the independent distortion model. For each model we state a random experiment for the generation of patterns based on an ideal structure. The results of applying these experiments to specific configurations is illustrated. Parameter estimation from a finite sample set and the scheme for computing the probabilities of blocks, obtained by applying the nonterminal rewriting rules, are also

presented. The estimation procedure for the Markov random field distortion model is quite complex, and we have presented an approximation to simplify this computation, that is applicable in certain cases. These noise and distortion models are quite general, and may be used in other image modelling applications. For example, the nonhomogeneous Markov random field model can be applied to texture analysis and the analysis of plant and crop data [8].

CHAPTER VI

EXPERIMENTAL RESULTS AND COMPUTATION

This chapter presents applications of the array grammar inference scheme. The two structures studied are the triangular patterns and the numeral '7', whose array grammars and ideal structures are presented in Figures 2 and 3 in Chapter 2. The training sets are versions of the ideal structures corrupted by three types of noise and distortion.

The first set of experiments uses the two dimensional discrepancy measure defined in Section 4.4 to compare the robustness of the inference algorithm. Robustness properties are further investigated by parsing experiments defined in Section 6.2. The second objective is to examine the effect of errors in the ideal configuration. A set of noisy patterns is generated after rotating the ideal structure slightly and the rate at which the array grammars inferred from the training sets generated by the three corrupting processes accept the samples is determined. The last section suggests the use of parallel techniques to reduce the computation time for the array grammar inference scheme.

6.1 The Inference Scheme

This section applies the inference scheme presented in Section 3.2 to training sets whose patterns are generated by independent noise, independent distortion and the Markov random field distortion models of Chapter 5. The triangular pattern structure, illustrated in Figure 2 (Chapter 2), was used for most of our experiments. The remaining experiments generate a p-array grammar inferred from noisy versions of the figure '7' (Fig. 3, Chapter 2). This illustrates the concept of coding matrix samples into strings using vector primitives.

The parameters used to generate each of the training sets are summarized in Table 2. All the training sets contained 20 patterns. With independent noise and distortion models the pattern types were chosen uniformly from the range [7,10]. With Markov random field distortion, the training set was made up of 10 triangles of type 9 and 10 triangles of type 10. This ensured a sufficient number of patterns of each size for parameter estimation. The training set for the '7's were either 8x8 or 11x11 arrays corresponding to pattern types 2 and 3, corrupted by the uniform noise model. The inference of the array grammar for the 7's takes an enormous amount of computer time and memory so we did not employ the distortion models with it. The parameters used to generate each of the training sets are summarized in Table 1. The probability of change parameters p, p_i and p_b were chosen so as to allow a reasonable amount of deviation without completely

destroying the underlying pattern structure. The value of p_b was set higher than p_i so as to allow greater distortions in the boundary than in interior regions. For the Markov random field model the parameter values were selected after a number of test runs. Parameters a_i and a_b in Table 2 introduce reasonable amounts of distortion into the patterns, with more changes likely in the boundary regions. Reasonable amounts of distortion along the three edges of the triangle and random changes are obtained in the interior regions for the values chosen for $a_{i,i}$, $a_{b,b}$ and $a_{b,i}$ in Table 2.

Table 2: Parameters values used to generate training sets

Generative model	Parameter values	Number of patterns
I) Pattern Structure: Triangles.		
1) Independent noise	p = 0.05, N U[7,10]	20
2) Independent distortion	p_i = 0.05, p_b = 0.15 N U[7,10]	20
3) Markov random field distortion	A = <2.2,1.5,0.1,-1.0,-0.8> N = 9,10	20
II) Pattern Structure: '7'		
1) Independent Noise	p = 0.05, N U[2,3]	20

N denotes the pattern type.

The first part of this section presents the results obtained at each stage of the inference scheme. The second part studies the robustness of the inference algorithm and demonstrates that the Markov random field model is quite general and covers the other two schemes.

6.1.1 Implementing the Inference Scheme

The first step in the inference scheme, described in Fig. 5 (Chapter 3), is to estimate the parameters of the model from the patterns of the training set as training set as described in Chapter 5. The noise/distortion process used to generate the training patterns is referred to as the generative model, whereas the model assumed when estimating the parameters for the inference scheme is termed the inferential model. With independent noise and distortion inferential models, this involved counting over the entire pattern set. However, the estimation procedure for the Markov random field model depends on size, so parameters were estimated separately for each size. Since the pattern sizes are small, the counts for all the patterns of one size are accumulated together for estimation. Our final estimates average the values obtained over all sizes and the two codings (Section 5.3.4). The estimates for each of the training sets appears in Table 3.

Table 3: Estimated Parameter Values

Generative Model	Parameter values
I) Pattern Structure: Triangles	
1) Independent noise	$p = 0.0468$
2) Independent distortion	$p_i = 0.0543$, $p_b = 0.137$
3) Markov random field distort.	A = <1.86,1.061,0.097,-1.166,-0.632>
II) Pattern Structure: '7'	
1) Independence Noise	$p = 0.051$

Step 2 in Fig. 5 breaks up the patterns into individual blocks, computes their probability, and separates them into their respective intermediate languages. For the '7's the blocks were coded into 'vector primitives' and converted to string form as shown in Fig. 16.

The k-tail method for inferring regular string grammars produces a number of candidate grammars corresponding to different tail lengths. To emphasize structural differences, weight factors $b_1 = 1.0$ and $b_2 = 10.0$ were used to compute the overall discrepancy (Section 4.2.3). The complexity and discrepancy values were plotted as functions of tail length and the grammar corresponding to the tail length that is closest

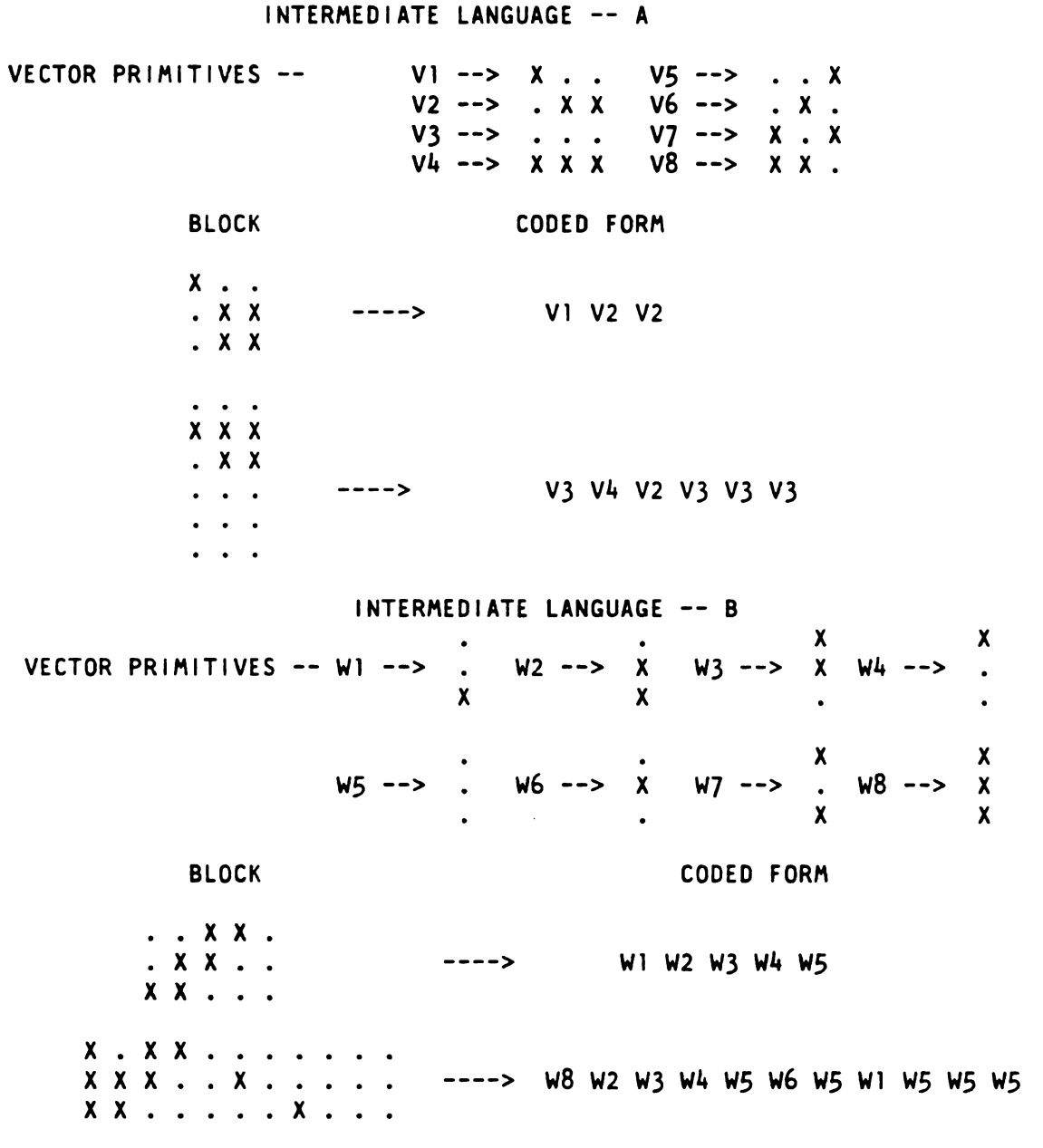


Fig. 16: Coding Blocks into Strings for Numeral 7

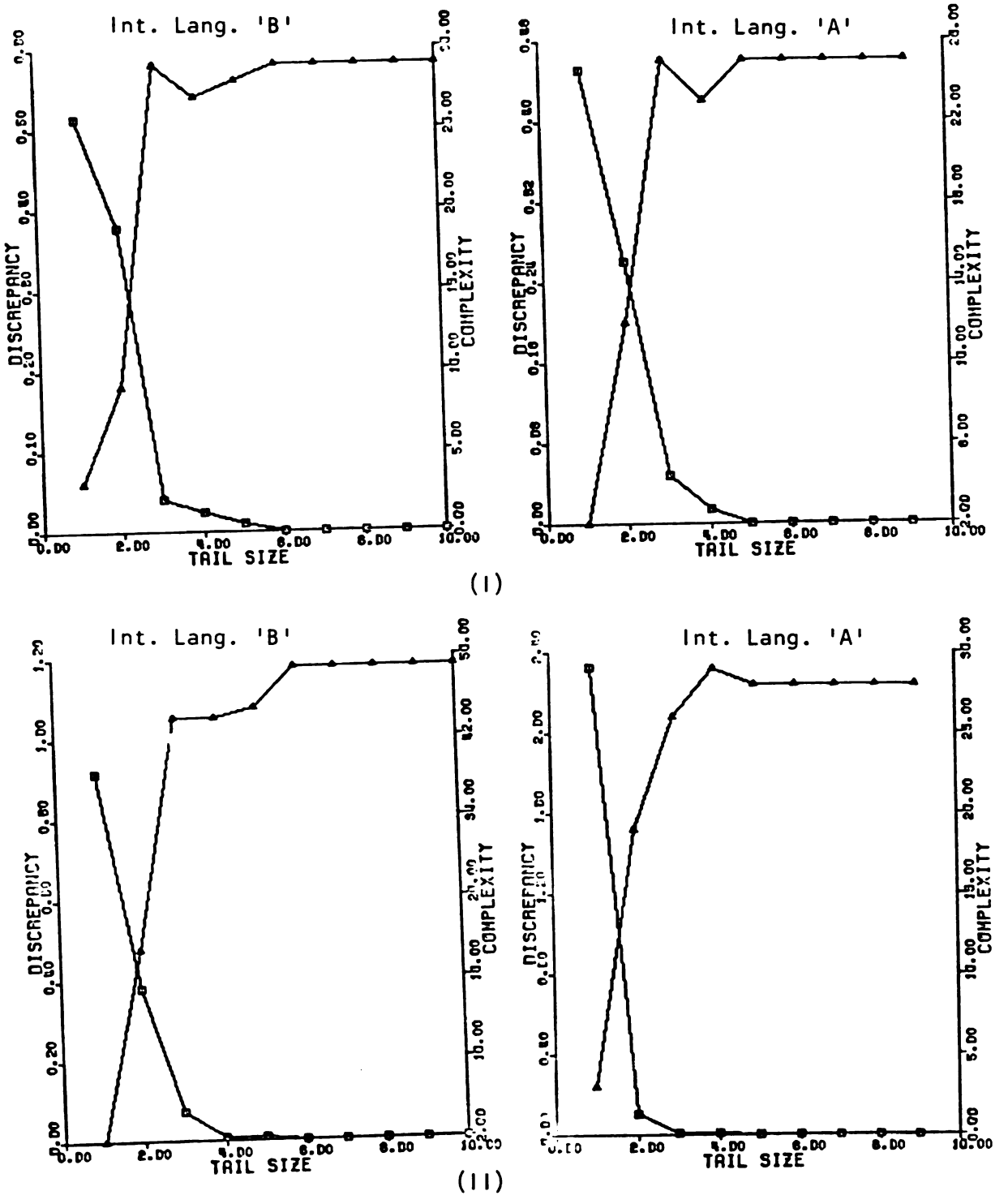
to the point of intersection of the two graphs was selected as the

'best' grammar for the intermediate language. Some of these plots are illustrated in Figure 17.

The inference procedure was implemented by a Spitbol (Snobol) program on the Cyber 170/750 computer. However, to reduce the total computation time, the discrepancy computation was implemented in a separate Fortran routine. The run time and the amount of core memory required by the inference program is presented in Table 4 for each of the training sets; L_A and L_B correspond to the two intermediate languages for each array grammar as illustrated in Figs. 2 and 3 (Chapter 2). The table indicates that the run time and core requirements for the inference algorithm are extremely high, even for the moderate pattern sizes used in our experiments. This was the main reason for limiting the number of experiments conducted.

6.1.2 Robustness Tests

The purpose of the robustness test is to assess the effectiveness of the inference procedure when the generative and inferential models are not the same. The two dimensional discrepancy measure assesses the fit of the inferred array language to the training set.



(I) Generative Model: Noise.

(II) Generative Model: MRF Distortion.

Fig. 17: Complexity and Discrepancy versus Tail length.

Table 4: Run Times and Core Requirements

Generative Model	Run Time (secs.) L_A	Run Time (secs.) L_B	Max. Core Used* (in K-words) L_A	Max. Core Used* (in K-words) L_B
Pattern Structure: Triangle				
1) Independent Noise	295.1	506.7	72.6	92.5
2) Independent Distort.	361.2	559.9	72.6	72.6
3) Markov Random Field	299.4	909.3	72.6	98.5
Pattern Structure: '7'				
1) Independent Noise	867.3	906.2	76.6	83.2

*- Each word on the Cyber is 60 bits long. The maximum core available to a user is 127.5 K-words.

The triangle was used as the ideal structure. Three training sets of 20 patterns each were formed; 10 of the patterns were of type 9 and the other 10 were of type 10. The parameters for the generation process were the same as in Table 2 and the corresponding array grammars were inferred in the usual way. The discrepancy measures are presented in Table 5.

Rows 1, 2 and 3 of Table 5, with the independent noise generative model, show that the Markov random field distortion model subsumes the

Table 5: Robustness Test - using the Discrepancy Measure

Generative Model	Inferential Model	Suitability Value
1) Indep. Noise	Indep. Noise	1.061 E-16
2) Indep. Noise	Indep. Distort.	2.090 E-16
3) Indep. Noise	MRF Distort.	0.245 E-16
4) Indep. Distort.	Indep. Noise	17.62 E-16
5) Indep. Distort.	Indep. Distort.	1.933 E-16
6) Indep. Distort.	MRF Distort.	0.661 E-16
7) MRF Distort.	Indep. Noise	37.91 E-16
8) MRF Distort.	Indep. Distort.	34.02 E-16
9) MRF Distort.	MRF Distort.	0.050 E-16

independent noise model. Rows 4, 5 and 6 show that with the independent distortion generative model, an inferential model of independent noise creates a high discrepancy whereas a Markov random field inferential model produces an even better fit than the "correct" inferential model. Comparing rows 3 and 6 with the discrepancy values obtained in rows 7, 8 and 9, emphasizes the generality of the Markov random field inferential model, which produces more suitable grammars than the other two inferential models create for the same generative model (rows 1 and 5). This justifies the use of the Markov random field inferential model.

6.2 Parsing Experiments

The two experiments reported in this section compare acceptance probabilities for triangles generated by the three different noise/distortion models. In the first experiment, the array grammars used for parsing were inferred in Section 6.1.1 (Table 3, part 1) and the patterns parsed were generated from the same array grammars. The second experiment created 15 distorted triangles from the physical processes themselves, five under each of the noise/distortion models. The array grammars used to parse them were inferred as in experiment 1 except that intermediate grammars of tail length one were selected. These array grammars are less restrictive than the grammars in experiment 1.

The results of experiment 1 are given in Table 6. Except for pattern number 14, the probability of acceptance for each pattern was highest when the pattern was parsed by the array grammar that generated it. Some of the patterns generated by one of the corruption processes could not be parsed by the array grammars trained on patterns from the other corruption processes which demonstrates differences among the array grammar models of Section 6.1.1.

The results of the second experiment are listed in Table 7. The grammar based on the Markov random field model accepted all 15 patterns, whereas grammars based on the independent models could parse

Table 6: Acceptance Probabilities - Patterns generated from Array Grammars

Source	Pattern Number	Size	Inferential Model INN	IND	MRF
INN	1	9	0.158 E-17	0.032 E-17	0.0001 E-17
INN	2	10	0.199 E-19	0.026 E-19	0.0001 E-19
INN	3	10	0.203 E-22	R	R
INN	4	11	0.681 E-23	0.026 E-23	R
INN	5	11	0.602 E-26	0.049 E-26	R
IND	6	9	0.104 E-19	0.207 E-19	R
IND	7	10	0.013 E-27	0.233 E-27	R
IND	8	10	R	0.424 E-26	0.179 E-26
IND	9	11	R	0.275 E-29	R
IND	10	11	0.178 E-31	0.512 E-31	R
MRF	11	9	R	R	0.629 E-25
MRF	12	10	R	0.009 E-30	0.716 E-30
MRF	13	10	R	R	0.747 E-32
MRF	14	11	0.017 E-31	0.327 E-31	0.004 E-31
MRF	15	11	R	R	0.353 E-35

INN - Independent noise model; IND - Independent distortion model;
MRF - Markov random field model.
R - Rejection; pattern could not be parsed by that grammar.

only two patterns generated under the Markov random field model. This illustrates that the Markov random field model generalizes both the other models. Table 7 also shows that the independent distortion model generalizes the independent noise model. In a few cases, the patterns were accepted with higher probability by an inferential model that was different from the generative model (e.g. patterns 1 and 3), but this is more likely a spurious result caused by the random nature of the generative process.

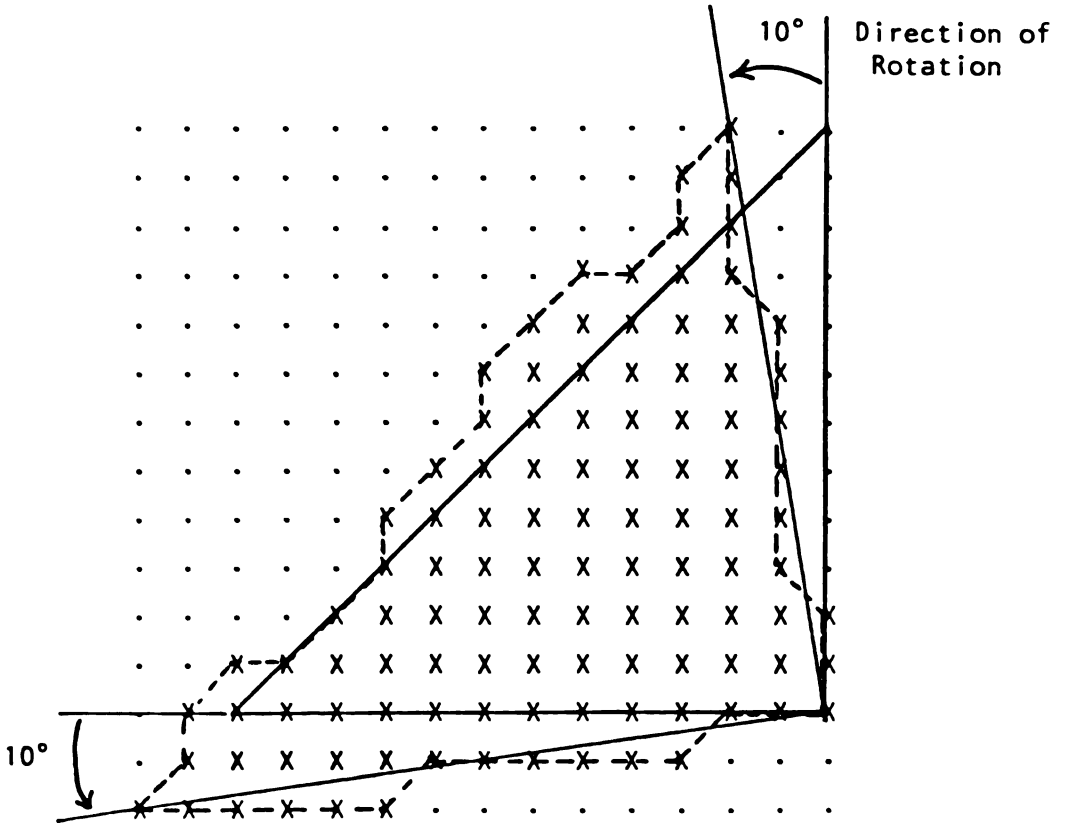
Table 7: Acceptance Probability - Patterns Generated from Physical process

Source	Pattern Number	Size	Inferential Model INN	IND	MRF
INN	1	6	0.051 E-11	0.176 E-11	0.086 E-11
INN	2	6	0.165 E-13	R	0.294 E-13
INN	3	11	0.354 E-33	0.530 E-33	0.014 E-33
INN	4	11	0.508 E-32	R	0.006 E-32
INN	5	12	0.191 E-30	0.015 E-30	< E-33
IND	6	6	0.016 E-13	0.193 E-12	0.308 E-12
IND	7	6	R	0.164 E-13	0.002 E-13
IND	8	12	0.009 E-38	0.801 E-38	0.074 E-38
IND	9	12	R	0.591 E-39	0.410 E-39
IND	10	12	R	R	0.266 E-43
MRF	11	12	R	R	0.512 E-45
MRF	12	12	0.0002 E-47	R	0.180 E-47
MRF	13	12	< E-51	R	0.141 E-47
MRF	14	12	R	R	0.488 E-46
MRF	15	12	R	R	0.144 E-52

INN - Independent noise model; IND - Independent distortion model;
MRF - Markov random field model.
R - Rejection; pattern could not be parsed by that grammar.
< - The acceptance probability was less than the value indicated.

6.3 Recognition of Rotated Triangles

The purpose of this experiment is to examine the effects of small rotations of the ideal configuration on the acceptance rate. A triangle of type 13 was rotated by 10 degrees as shown in Figure 18. Ten patterns were created by imposing the independent noise model with a probability of change value of 0.05 on the rotated triangles and parsed by the three array grammars of Section 6.1.1; the results are



Boundary of original 13x13 triangle.

Boundary of rotated triangle.

Fig. 18: Rotated Triangle.

presented in Table 8. Some of the noisy rotated triangles are shown in Figure 19.

The Markov random field model grammar accepted 9 of the 10 rotated patterns though the probability of acceptance was quite low (the acceptance value is of the order of E-47 for the ideal triangular structure of size 15). The noise and distortion model grammars failed to recognize these triangles. This shows that an inference scheme based on the Markov random field distortion model can incorporate small variations in the ideal structure into the distortion process.

Table 8: Acceptance Probabilities - Rotated Triangles

Source	Pattern Number	Inferential Model INN	IND	MRF
INN	1	R	R	0.132 E-97
INN	2	R	R	0.408 E-104
INN	3	R	R	0.482 E-93
INN	4	R	R	0.262 E-92
INN	5	R	R	0.725 E-95
INN	6	R	R	R
INN	7	R	R	0.388 E-97
INN	8	R	R	0.434 E-101
INN	9	R	R	0.275 E-92
INN	10	R	R	0.723 E-97

INN - Independent noise model; IND - Independent distortion model;
MRF - Markov random field model.
R - Rejection; pattern could not be parsed by that grammar.

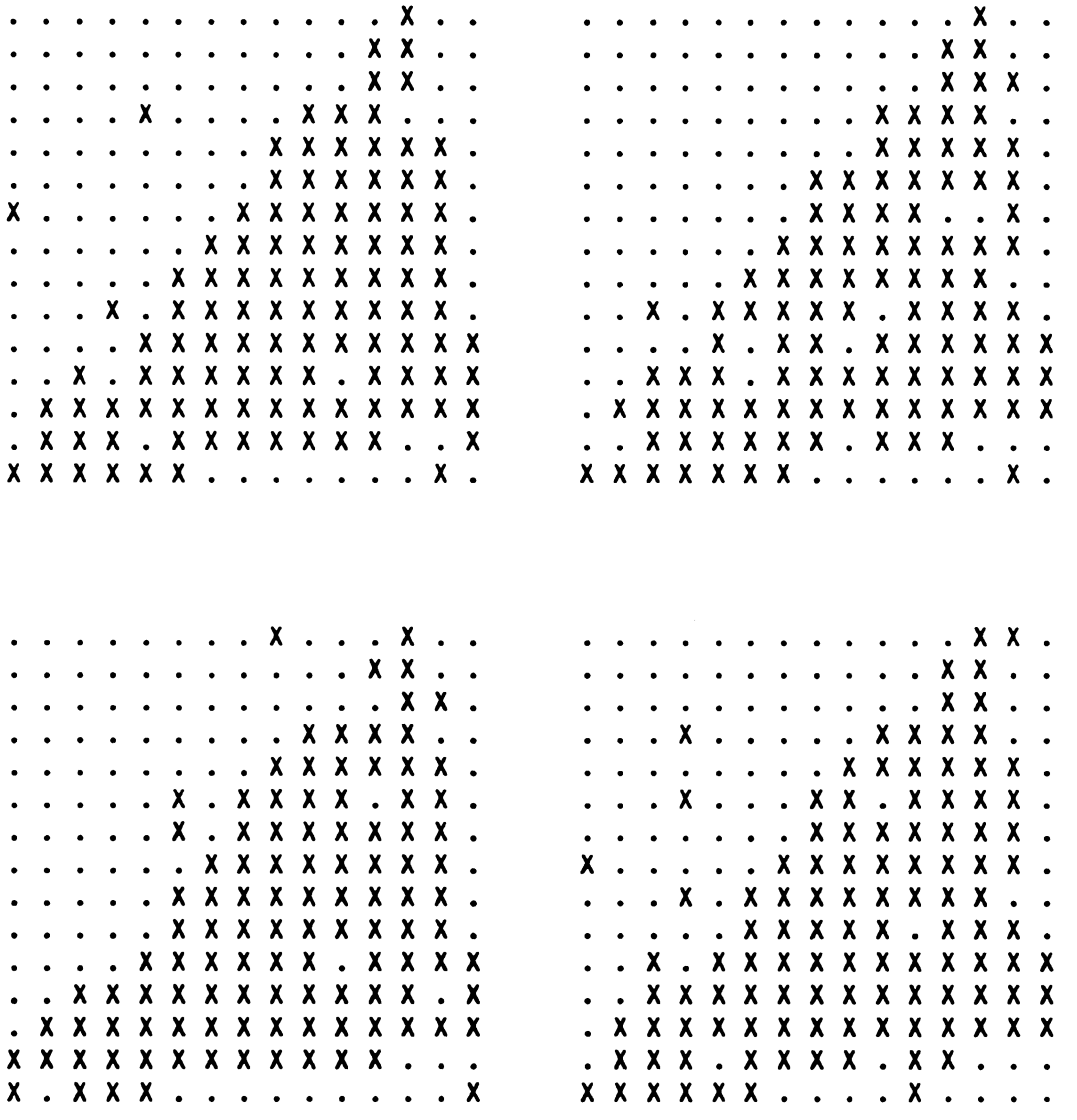


Fig . 19: Noisy Rotated Triangles

6.4 Parallel Computation Techniques

The inference algorithm for regular grammars, the algorithm for computing the discrepancy measure and the computation of the measure of suitability for array languages require large amounts of computer time in the normal sequential mode of operation. However, parallel computation techniques, such as the SIMD (Single Instruction-Multiple Data) and pipeline architectures, offer significant speedup. Parallel algorithms have been presented for the k-tail inference scheme for string grammars [16] and the computation of the WLD between two strings [16,17], which can be implemented on general purpose pipeline processors [6] or dedicated SIMD systems [17]. The time complexity for the WLD computation for two strings of lengths m and n is reduced from O(mn) to O(m+n), but requires [min(m,n)+1] processing elements [17].

We now extend the parallel algorithm for computing the WLD for strings to a parallel algorithm for computing WLD2, the distance between two arrays. This algorithm, a parallel implementation of the algorithm from Moore [76], is presented below. The notation WLD(COL-J,COL-L) denotes the WLD measure for strings between column J of the IxJ subarray from A and the L^{th} column from the KxL subarray from B. WLD(ROW-I,ROW-K) has a similar interpretation. SUBS(A_{PQ},B_{RS}) represents the substitution cost of replacing element A_{PQ} of array A by element B_{RS} of array B. It is 0 if $A_{PQ} = B_{RS}$, and SC (the substitution cost) otherwise.

Procedure 4:

Input: Two arrays, A and B, of sizes PxQ and sizes RxS respectively.

SC, the substitution cost, IC, the insertion cost and DC, the deletion cost.

Begin

(1) Set Dist(0,0,0,0) = 0

(2) For M = 1 to P+Q+R+S

Do in Parallel For all I,J,K,L = 0 to M such that I+J+K+L = M, steps (i)-(xvi)

(i) e_1 = Dist(I,J,K,L-1) + K*IC

(ii) e_2 = Dist(I,J,K-1,L) + L*IC

(iii) e_3 = Dist(I,J-1,K,L) + I*DC

(iv) e_4 = Dist(I-1,J,K,L) + J*DC

(v) e_5 = Dist(I,J,K-1,L-1) + (K*L-1)*IC

(vi) e_6 = Dist(I-1,J-1,K,L) + (I*J-1)*DC

(vii) e_7 = Dist(I,J-1,K,L-1) + WLD(COL-J,COL-L)

(viii) e_8 = Dist(I-1,J,K-1,L) + WLD(ROW-I,ROW-K)

(ix) e_9 = Dist(I,J-1,K-1,L) + L*IC + I*DC

(x) e_{10} = Dist(I-1,J,K,L-1) + K*IC + J*DC

(xi) e_{11} = Dist(I,J-1,K-1,L-1) + (J-1)*DC + WLD(COL-J,COL-L)

(xii) e_{12} = Dist(I-1,J,K-1,L-1) + (K-1)*IC + WLD(ROW-I,ROW-K)

(xiii) e_{13} = Dist(I-1,J-1,K,L-1) + (J-1)*DC + WLD(COL-J,COL-L)

(xiv) e_{14} = Dist(I-1,J-1,K-1,L) + (I-1)*DC +

WLD(ROW-I,ROW-K)

(xv) e_{15} = Dist(I-1,J-1,K-1,L-1) + WLD(ROW-I,ROW-K) +

WLD(COL-J,COL-K) - SUBS(A_{PQ},B_{RS})

(xvi) Dist(I,J,K,L) = min(e_i; 1<=i<=15).

End Loop

End Loop

Distance(A,B) = Dist(P,Q,R,S)

End Procedure.

At each step in the WLD2 computation, the string WLD between strings of length I and K, and J and L respectively, is computed. Therefore, the WLD2 computation for PxQ and RxS arrays requires P+Q+R+S steps with a time complexity of $O\{(P+Q)+(R+S)\}$ at each step. The overall time complexity can then be expressed as

$$O\{(P+Q+R+S)^2\}$$

For P=Q=R=S=N this reduces to $O(N^2)$, as opposed to $O(N^6)$ for the sequential case. The general expression for the maximum number of processors required is extremely complicated, but for I=J=K=L=N can be expressed as

$$\frac{(N+1)\ (2N^2+4N+3)}{3} \quad , \text{ for } N>=2 \qquad --(1)$$

For N = 1, seven processors are required.

This shows that the number of processors required is of the order of N^3. The WLD2 computation of two 20x20 arrays requires 6181 processors, a rather large number. A more practical scheme might involve using a

smaller number of processors, say U, and a pipeline architecture. Ignoring the delay in the set-up time, or the time taken to structure the pipeline for the computation, the number of time steps required for computation can be expressed as,

$$\sum_{i=1}^{I+J+K+L} \left\lceil \frac{M_i}{U} \right\rceil$$

where M_i is the number of processors required at each step (2) of the above algorithm. For I=J=K=L=N, M_i can be expressed recursively as

$$M_i = \frac{(i+1)(i+2)(i+3)}{6} \qquad 1 \le i \le N$$

$$= M_{i-1} + i+1 \qquad i = N+1$$

$$= M_{i-1} + (2N-1) - 3(i-N-1) \qquad N+2 \le i \le 2N$$

$$= M_{4N-i} \qquad 2N < i \le 4N-1$$

Therefore, the use of pipeline architectures results in fairly large speedup without requiring an unreasonable number of processing elements, even for the WLD2 computation between large arrays. For example, the WLD2 computation between two 128x128 arrays would require an enormous number of processors (4,293,507) by equation 1 for a SIMD implementation. The use of a pipeline architecture with 4096 processors will increase the number of computation steps from $O(N^2)$ to $O(N^{3.06})$, which is still a considerable improvement over the $O(N^6)$ steps required for a sequential scheme. However, the actual speedup obtainable in a

practical implementation will also depend on memory bandwidth and the interconnection network used between memory and the processors.

6.5 Summary

In this chapter, we have presented results obtained by applying the inference scheme in conjunction with the independent noise, the independent distortion and the Markov random field distortion models. The discrepancy measure for array languages and parsing experiments show that the inference scheme based on the Markov random field model is more robust than those based on the other two schemes. The acceptance rates for the set of rotated triangles shows that the Markov random field model introduces flexibility into the syntactic recognition scheme by allowing small deviations and changes in the ideal structure at the cost of increased computational complexity. The last section describes parallel techniques that may be used to speed up the computation process.

CHAPTER VII

SUMMARY, CONCLUSIONS AND FUTURE RESEARCH

This chapter summarizes the main contributions of this thesis and suggests directions for future research.

7.1 Summary and Conclusions

The goal of this thesis was to develop a two dimensional picture description and learning scheme for binary images using the array grammar as a formal mathematical model. The block structure of array grammars has been exploited to develop an inference scheme that requires knowledge of the ideal image, which implies the nonterminal rewriting rules be known. Chapter 3 of this thesis proposes the first formal algorithm for inferring an array grammar that models the structure of noisy and distorted patterns. The key step is to infer intermediate grammars from probabilistic samples. The development of the probabilistic two dimensional inference scheme is the primary contribution of this thesis. Several of the subproblems that were attacked are summarized below.

A number of complexity and discrepancy measures for string grammars are reviewed in Chapter 4, and a computationally simple complexity measure that permits quantification of the differences in complexity among regular grammars inferred by the k-tail method was chosen for our application. The main development in this part of the thesis is the definition of a new discrepancy measure for string languages in Section 4.3 that involves both probabilistic and structural factors. The new discrepancy measure is shown to be bounded and convergent, making it computationally feasible, even for infinite languages. It is also shown to possess a number of intuitively desirable properties. The concept of the structural discrepancy measure for string languages is extended to define a two dimensional discrepancy measure for array languages in Section 4.4 that quantifies the ability of an inferred array language to match the training set.

Chapter 5 discusses a simple independence noise model and two independent distortion models for describing corrupting influences on images. A powerful and comprehensive distortion model based on nonhomogeneous Markov random fields is also analyzed. These models have wider applications than grammatical inference. Mathematical formulations and techniques for generating patterns are studied. Estimation of the parameters of each model are defined, given a finite set of training patterns. The estimates are used to compute the probabilities of the block samples for each intermediate language.

Tests performed on isosceles triangles indicate that the array grammars inferred under the Markov random field models are capable of accepting patterns that are generated by independent noise and distortion processes. Approximations to the two dimensional discrepancy measure of Section 4.4 have been used to investigate the robustness of array grammars that are inferred from three generative models. The Markov random field inferential model was found to be the most robust in Section 6.1.2. Parsing experiments in Section 6.2 demonstrated the capability of the Markov random field inferential model to parse patterns from other models. Results of an experiment on a rotated triangle in Section 6.3 show that the array grammar inferred under the Markov random field inferential model is capable of parsing noisy rotated patterns. The two main conclusions of this study are first, the Markov random field model infers a versatile two dimensional grammar that is applicable with a number of noise and distortion models, and second, it allows the use of underlying structures that have small deviations from the original ideal structure, thus making the inferential model more flexible.

The run time and the memory requirements for the inference algorithm on the Cyber 170/750 computer show that the proposed algorithm requires tremendous amount of computation time in the sequential mode of operation. The lack of adequate computer resources was the main reason for the limited number of experiments conducted in

this thesis. The array grammar model is more suited for a sequential/parallel mode of operation. Chapter 6 examines potential computational speedup obtainable by using SIMD and pipeline architectures. A parallel implementation of the WLD2 algorithm is proposed, that reduces the time complexity of the algorithm from $O(N^6)$ to $O(N^2)$. Possible implementation using a pipeline architecture when the number of processing elements required becomes too large, is also discussed.

The study of a two dimensional learning scheme in conjunction with a corrupting scheme is the primary contribution of this thesis. The main conclusion is that the learning scheme based on the Markov random field model infers a vesatile array grammar that is applicable with a number of noise and distortion models and also allows small deviations in the assumed ideal structure. Implementation of this scheme with the architecture proposed in Section 6.4 should open up a number of real time practical applications in industrial vision and robotics.

7.2 Future Research

Directions for further research can be classified into two categories -

(i) Modifications and extensions to the inference scheme and the noise and distortion models that enhance their performance, and

(ii) Implementation of cellular automata as language acceptors [103] and of parallel rewriting rules [59,94], to define a new form of formal

array grammar model that could overcome the computational problems of present two dimensional inference schemes.

A few problems from category (i) are outlined below. An important factor that arises after the matrix samples have been converted into vector primitives is the size of the set of primitives. Formal primitive extraction procedures could be studied using the overall complexity of the inferred grammar as a criterion. The grammar would be accepted as a candidate grammar only if its complexity was below a prespecified threshold. In case no string grammar could be found satisfying the above criterion, an alternative scheme that involves inferring a two level matrix grammar(Siromoney) could be adopted, as is explained in Appendix C. When the underlying noise/distortion model for the sample set is not known, statistical tests might be developed to model the noise and distortion in the training set. A decision making procedure would be required to choose the appropriate model. This procedure could be based on computing the difference in probabilities assigned to samples by the distortion scheme, and the apriori probabilities assigned to the sample. The model that provides the closest fit could then be chosen as the underlying distortion scheme for the pattern class. This scheme could be further advanced to modelling the noise and distortions observed in real image processing applications.

Another direction for future research, involves trying to fit the concepts of cellular automata as language acceptors [103,104,105] and the concept of parallel rewriting rules [59,94,101,102] into the array grammar model. Intuitively, a cellular automaton is an array of identical processing elements called cells, which are uniformly connected in some form of a neighborhood pattern. Each cell is connected in a particular configuration to a fixed number of neighboring processors. These cells operate synchronously and in discrete time steps. Reconfigurable cellular processors have also been defined [30]. Cellular computers are presently being used for image processing operations [32,106]. The usefulness of cellular automaton as language acceptors and their ability for fast parsing of regular and context free languages indicate possible applications in inference and learning schemes. The concept of parallel two dimensional parsing using cellular architectures also indicates new possibilities in modelling the structures of two dimensional images and developing learning schemes for the same.

APPENDICES

APPENDIX A

NOTATION AND DEFINITIONS - ARRAY GRAMMARS

We define I to be an alphabet - a finite non empty set of terminal symbols. A matrix over I is an mxn array of symbols from I. The set of all matrices over I is denoted by I^{**} and $I^{++} = I^{**} - \{\ \}$, where the empty matrix is denoted Λ. If

$$X = \begin{bmatrix} a(1,1) & a(1,2) & \cdots & a(1,n) \\ a(2,1) & a(2,2) & \cdots & a(2,n) \\ \cdots & \cdots & & \cdots \\ \cdots & \cdots & & \cdots \\ a(m,1) & a(m,2) & \cdots & a(m,n) \end{bmatrix} \qquad Y = \begin{bmatrix} b(1,1) & b(1,2) & \cdots & b(1,n') \\ b(2,1) & b(2,2) & \cdots & b(2,n') \\ \cdots & \cdots & & \cdots \\ \cdots & \cdots & & \cdots \\ b(m',1) & b(m',2) & \cdots & b(m',n') \end{bmatrix}$$

are two matrices then column catenation is defined only when $m = m'$, and is written as,

$$X \phi Y = \begin{bmatrix} a(1,1) & a(1,2) & \cdots & a(1,n) & b(1,1) & \cdots & b(1,n') \\ \cdots & \cdots & & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & & \cdots & \cdots & \cdots & \cdots \\ a(m,1) & a(m,2) & \cdots & a(m,n) & b(m,1) & \cdots & b(m',n') \end{bmatrix}$$

The new matrix has m rows and (n+n') columns. Similarly row catenation is defined only when n = n', and is written as,

$$X \ \theta \ Y = \begin{bmatrix} a(1,1) & \text{-----} & a(1,n) \\ \text{----} & \text{----} & \text{----} \\ \text{----} & \text{----} & \text{----} \\ a(m,1) & \text{-----} & a(m,n) \\ b(1,1) & \text{-----} & b(1,n) \\ \text{----} & \text{----} & \text{----} \\ b(m',1) & \text{----} & b(m',n) \end{bmatrix}$$

The resultant matrix has (m+m') rows and n columns. If M and M' are two sets of matrices, then the column and row products are sets of matrices defined as follows

Column Product, $M \ \phi \ M' = \{X \ \phi \ Y \mid X \in M, Y \in M'\}$

and Row Product, $M \ \theta \ M' = \{X \ \theta \ Y \mid X \in M, Y \in M'\}$.

Columnwise Kleene closure is defined iteratively as follows:

Let - $M^1 = M$, $M^2 = M^1 \ \phi \ M$, ---, $M^{n+1} = M^n \ \phi \ M$.

$$M^+ = \bigcup_{i>=1} M^i$$

$$M^* = M^+ \cup (\Lambda)$$

Kleene closure by rows is defined as

$$\text{Let} - M_1 = M,\ M_2 = M_1 \ \theta\ M, \text{-------},\ M_{n+1} = M_n \ \theta\ M.$$

$$M_+ = \bigcup_{i>=1} M_i$$

$$M_* = M_+ \ U\ (\Lambda)$$

APPENDIX B

ARRAY AUTOMATA

An array automaton consists of a two dimensional input tape and a two dimensional storage tape. If the input is an mxn array of cells, the storage tape contains a $(2m+1)\times(2n+1)$ array of cells. The movement of the automaton is in three stages. In the first, the automaton acts on the storage tape, reading, printing and "pushing" symbols around depending on the nonterminal production rules, ultimately subdividing the $(2m+1)\times(2n+1)$ storage tape into blocks, each of which corresponds to a matrix of a corresponding IML. In the second step, the automaton acts as several automata and fills up each symbol (i,j) of the storage tape ; i and j are both even numbers. The blocks can have a fixed number (say $k >= 1$) of rows (or columns) so the automaton must act on these rows or columns simultaneously. The third step consists entirely of checking. The automaton compares the input symbol in the (i,j)th cell with the symbol on the $(2i,2j)$th cell of the storage tape. If the comparison fails at any cell the automaton halts without accepting. Otherwise the automaton halts and accepts the input.

The set of patterns accepted by an array automata are equivalent to the set of arrays generated by array grammars [60]. We conclude this section with a formal definition of a (regular:regular) array automaton

{(R:R) AA}.

The (regular:regular) array automaton is a 10 tuple

$$(K, I, Z, d_1, d_2, d_3, q_0, Z_0, F, C)$$

where

K is a set of finite states;

$q_0 \in K$ is a distinguished (start) state;

$F \subset K$ is the set of final states;

I is the input alphabet, or the primitive picture symbols;

$Z = Z_1 \cup Z_2 \cup Z_3$, with $Z_3 = I$ is the finite set of storage symbols;

$Z_0 \in Z_1$ is the initial storage symbol;

d_1 is a mapping from $\{q_0\} \times Z_1$ into finite subsets of $\{q_0\} \times \{Z_1 \phi Z_2\}$ or $\{q_0\} \times \{Z_2 \phi Z_1\}$ where ϕ represents row catenation (θ) or column catenation (ϕ);

d_2 is a mapping from $K \times Z_2$ into the finite subsets of $K \times \{(Z_2 \phi Z_3) \cup (Z_3 \phi Z_2)\}$;

d_3 is a mapping from $K \times I \times I$ into $K \times \{DOWN, TOP\}$;

C is a counter that starts from an initial value of 0, and keeps track of the d_1 moves of the automaton.

The d_1 moves of the automaton correspond to the nonterminal rewriting rules of the corresponding array grammar, and create the block structure description of the required size. The d_2 moves are the second step and correspond to filling each block independently with the primitive symbols of the picture. The d_3 mapping corresponds to the checking phase, where the input picture is compared to the picture on

the storage tape. DOWN implies a move downward along a column and TOP implies a change to the top of the next column.

APPENDIX C

MATRIX GRAMMARS

A matrix grammar is a two tuple

$$MG = (G_1, G_2)$$

where

$G_1 = (V_1, I_1, P, S)$ is a string grammar that can be context-sensitive, context-free or regular;

V_1 is a finite set of horizontal nonterminals;

I_1 is a finite set of intermediates $\{S_1, S_2, ---, S_k\}$;

P_1 is a finite set of production rules;

$S \in V_1$ is the start symbol;

$G_2 = \bigcup_{i=1}^{k} G_{2i}$ is the set of vertical grammars where each

$$G_{2i} = (V_{2i}, I, P_{2i}, S_i)$$

is a right linear (regular) grammar and

V_{2i} is a finite set of vertical nonterminals,

P_{2i} is a finite set of right linear production rules,

I is the set of picture primitives and

$V_{2i} \cap V_{2j} = \phi$ if $i \neq j$.

Derivations to create an mxn picture follow a two step procedure. First a string $s_1 s_2 --- s_n$ symbols from I_1 is generated horizontally

using G_1. Then the corresponding G_2 rules are applied m times simultaneously to every column, to create an mxn matrix. The set of all matrices generated by a matrix grammar is denoted

$$L(MG) = \{mxn \text{ arrays } [a_{ij}],\ m,n \geq 1 \mid S = s_1 s_2 \cdots s_n \Rightarrow [a_{ij}]\}.$$

LIST OF REFERENCES

REFERENCES

[1] A.V. Aho and T.G. Peterson, 'A minimum distance error-correcting parser for context free languages', SIAM Jour. Computing, vol. 4, 1972, pp. 304-312.

[2] J.E. Albus, 'Electrocardiogram intrepretation using a stochastic finite state model' in Syntactic Pattern Recognition: Applications (K.S. Fu, ed.), Springer Verlag, Berlin Heidelburg, 1977, pp. 51-64.

[3] F. Ali and T. Pavlidis, 'Syntactic recognition of handwritten numerals', IEEE Trans. Sys., Man and Cyber., vol. SMC-7, 1977, pp. 537-541.

[4] M.R. Anderberg, Cluster Analysis For Applications, Academic Press, New York, 1973.

[5] A. Apostolico, 'On the efficiency of syntactic feature extraction and approximation schemes for data compression', Proc. 5th. Intl. Conf. on Pattern Recognition, Miami Beach, Florida, vol. 2, 1980, pp. 982-984

[6] J.L. Baer, Computer Systems Architecture, Chapter 10.2, Computer Science Press, Maryland, 1980.

[7] J.E. Besag, 'Nearest neighbor systems and the auto-logistic model for binary data', J. Royal Stat. Soc., Ser. B, vol. 34, 1972, pp. 75-83.

[8] J.E. Besag, 'Spatial interaction and the statistical analysis of lattice systems', J. Royal Stat. Soc., Ser. B, vol. 36, 1974, pp. 192-236.

[9] A.W. Biermann and J.A. Feldman, 'On the synthesis of finite state machines from samples of their behavior', IEEE Trans. Comput., vol. C-21, 1972, pp. 592-597.

[10] M. Blum, 'On the size of machines', Info. and Control, vol. 11, 1967, pp. 257-265.

[11] T.L. Booth and R.A. Thomson, 'Applying probabilistic measures abstract languages', IEEE Trans. Comput., vol. C-22, 1973, pp. 442-450.

[12] J.R. Bourne, V. Jagannathan, B. Giese and J.W. Ward, 'A software

system for syntactic analysis of EEG', Computer Programs in Biomedicine, vol. 11, 1980, pp. 190-200.

[13] W.S. Brainerd, 'Tree-generating regular systems', Info. and Control, vol. 14, 1969, pp. 217-231.

[14] J.M. Brayer and K.S. Fu, 'A note on the k-tail method of tree grammar inference', IEEE Trans. Sys., Man, Cyber., vol. SMC-7, 1977, pp. 293-300.

[15] J.M. Brayer, 'Web automata and parsing web grammars', Report NSF Workshop on Structural and Syntactic Pattern Recog., Saratoga Springs, New York, 1981, pp. 40-41.

[16] Y. Chiang and K.S. Fu, 'Parallel processing of some synatctic pattern recognition algorithms', Ninth Workshop Appl. Imagery, Maryland, 1980, pp. 1-21.

[17] Y. Chiang and K.S. Fu, 'Parallel processing for distance computation in syntactic pattern recognition', IEEE Comput. Soc. Workshop Comput. Architecture for Pattern Analysis and Image Database Management, VA, 1981, pp. 337-344.

[18] N. Chomsky, Syntactic Structures, Mouton and Co., The Hague, Netherlands, 1957.

[19] N. Chomsky and G.A. Miller, 'Finite state languages', Info. and Control, vol. 1, 1958, pp. 91-112.

[20] V. Claus,et al., eds., Graph Grammars and their Applications to Computer Science and Biology, Springer Verlag, Berlin, Heidelberg, 1979.

[21] C.M. Cook, 'A cost function for concept formation', Tech. Rep. TR-212, Computer Science Center, Univ. of Maryland, College Park, 1972.

[22] C.M. Cook, 'Grammatical inference by heuristic search', Tech. Rep. TR-287, Computer Science Center, Univ. of Maryland, College Park, 1974.

[23] C.R. Cook and P.S. Wang, 'A Chomsky hierarchy of isotonic array grammars and languages', Computer Graphics and Image Processing, vol. 8, 1978, pp. 144-152.

[24] S. Crespi Reghizzi, 'The mechanical acquisition of precedence grammars', Tech. Rep. UCLA-ENG-7054, School of Engg. and Appl. Science, Univ. of California, Los Angeles, 1970.

[25] G.R. Cross, 'Markov random fields and texture models', Ph.D.

thesis, Michigan State Univ., E. Lansing, 1980.

[26] G.R. Cross, et al., 'Multivariate normality in Pattern Recognition and clustering', Proc. 6th. Intl. Conf. on Pattern Recog., Munich, Germany, 1982, pp. 862-864.

[27] R. DeMori, et al., 'A syntactic procedure for the recognition of glottal pulses in continuous speech', Pattern Recognition, vol. 9, 1977, pp. 181-189.

[28] P.J. Denning, et al., Machines, Languages and Computation, Prentice- Hall, New Jersey, 1978.

[29] R.L. Dobrushin, 'Gibbsian random fields for lattice systems with pairwise interactions', Funcl. Analy. Applic., vol. 2, 1968, pp. 292-301.

[30] T. Dubitzsky, et al., 'Parallel region property computations by active quadtree networks', IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-3, 1981, pp. 626-633.

[31] R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis, John Wiley, New York, 1973.

[32] M.J.B. Duff, 'Parallel processing techniques', from Pattern Recognition - Ideas in Practice (B.G. Batchelor, ed.), Plenum Press, 1978, pp. 145-176.

[33] J. Earley, 'An efficient context free parsing algorithm', CACM, vol. 13, 1970, pp. 94-52.

[34] J.A. Feldman et al., 'Grammatical complexity and inference', Tech. Rep. CS-125, Computer Science Dept., Stanford Univ., Stanford, 1969.

[35] J.A. Feldman, 'Some decidability results on grammatical inference and complexity', Info. and Control, vol. 20, 1972, pp. 244-262.

[36] N.V. Findler and J.V. Leeuven, 'A family of similarity measures between strings', IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-1, 1979, pp. 116-118.

[37] P.A. Flinn, 'Monte Carlo calculation of phase separation in a 2-dimensional Ising system', J. Stat. Physics, vol. 10, 1974, pp. 89-97.

[38] H. Freeman, 'On the encoding of arbitrary geometric patterns' IRE Trans. on Electronic Comp., vol. EC-10, 1961, pp. 260-268.

[39] K.S. Fu and T. Huang, 'Stochastic grammars and languages', Intl. Jour. Comp. and Info. Sciences, vol. 1, 1972, pp. 135-170.

[40] K.S. Fu and B.K. Bhargava, 'Tree systems for syntactic pattern recognition', IEEE Trans. Computers, vol. C-22, 1977, pp. 1087-1099.

[41] K.S. Fu, Syntactic Methods in Pattern Recognition, Academic Press, New York, 1974.

[42] K.S. Fu and T.L. Booth, 'Grammatical inference: introduction and survey', Part I, IEEE Trans. Sys., Man, Cyber., vol. SMC-5, 1975, pp. 95-111.

Part II, IEEE Trans. Sys., Man, Cyber., vol. SMC-5, 1975, pp. 409-423.

[43] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York, 1972.

[44] L.W. Fung and K.S. Fu, 'Stochastic syntactic decoding for pattern classification', IEEE Trans. Comput., vol. C-24, 1975, pp. 662-667.

[45] R.C. Gonzalez and M.G. Thomason, 'Tree grammars and their applications to pattern recognition', Tech. Rep. no. TR-EE/CS-74-20, Electrical Engineering Department, Univ. of Tennessee, Knoxville, 1974.

[46] R.C. Gonzalez and M.G. Thomason, Syntactic Pattern Recognition: an Introduction, Addison-Weseley, Reading, Mass., 1978.

[47] R.C. Gonzalez and R. Saafabakhsh, 'Computer vision techniques for industrial applications and robot control', Computer, vol. 15, 1982, pp. 17-32.

[48] J. Gruska, 'Some classifications of context free languages', Info. and Control, vol. 14, 1969, pp. 152-179.

[49] J.M. Hammersley and D.C. Handscomb, Monte Carlo Methods, Meuthen, London, 1964.

[50] J.M. Hammersley and P. Clifford, 'Markov random fields on finite graphs and lattices', (Unpublished manuscript).

[51] J. Hartmanis and R.E. Stearns, 'On the computational complexity of algorithms', Trans. American Math. Soc., vol. 117, 1965, pp. 285-306.

[52] J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory,

Languages, and Computation, Addison Weseley, Mass., 1979.

[53] J.J. Horning, 'A study in grammatical inference', Ph.D. Thesis, Stanford Univ., Stanford, CA, 1969.

[54] E. Isaacson and H.B. Keller, Analysis of Numerical Methods, John Wiley, New York, 1966.

[55] V. Isham, 'An introduction to spatial point processes and Markov random fields', Intl. Stat. Review, vol. 49, 1981, pp. 21-43.

[56] S.N. Jayaramamurthy, 'Multilevel array grammars for generating texture scenes', Proc. IEEE Conf. on Pattern Recognition and Image Processing, Chicago, Illinois, 1979, pp. 391-398.

[57] R.L. Kashyap and M.C. Mittal, 'Recognition of spoken words and phrases in multitalker environment using syntactic methods', IEEE Trans. Computers, vol. C-27, 1978, pp. 442-451.

[58] R.A. Kirsh, 'Computer intrepretation of english text and picture patterns', IRE Trans. Electronic Computers, vol. EC-13, 1964, pp. 363-376.

[59] H.C.M. Kleijn and G. Rozenberg, 'A study in parallel rewriting systems', Info. and Control, vol. 44, 1980, pp. 134-163.

[60] K. Krithivasan and R. Siromoney, 'Array automata and operations on array languages', Intl. Jour. of Computer Math., vol. 4, Sec. A, 1974, pp. 3-30.

[61] W. Kuich, 'On the entropy of context free languages', Info. and Control, vol. 16, 1970, pp. 173-200.

[62] R.S. Ledley, et al., 'FIDAC: Film input to digital automatic computer and associated syntax-directed pattern recognition programming system', Optical and Electro-Optical Info. Proc. (J.T. Tippet et. al., eds.), MIT Press, Cambridge, Mass., 1965, pp. 591-613.

[63] H.C. Lee and K.S. Fu, 'A stochastic syntax analysis procedure its application to pattern classification', IEEE Trans. Computers, vol. C-21, 1972, pp. 660-666.

[64] V.I. Levenshtein, 'Binary codes capable of correcting deletions, insertions and reversals', Sov. Phys. Dokl., vol. 10, 1966, pp. 707-710.

[65] B. Levine, 'Derivatives of tree sets with applications to grammatical inference', IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-3, 1981, pp.285-293.

[66] J. Liou, 'Grammatical inference by a constructive method', Ph.D. dissertation, Dept. of Computer Science, Michigan State Univ., E. Lansing, 1977.

[67] S.Y. Lu and K.S. Fu, 'Stochastic error-correcting syntax analysis for recognition of noisy patterns', IEEE Trans. Comput., vol. C-26, 1977 pp. 1268-1276.

[68] S.Y. Lu and K.S. Fu, 'A syntactic approach to texture analysis', Computer Graphics and Image Processing, vol. 7, 1978, pp.303-330.

[69] F.J. Maryanski, 'Inference of probabilistic grammars', Ph.D. dissertation, Dept. of Elec. Engg. and Computer Science, Univ. of Connecticut, Storrs, 1974.

[70] K.V. Mardia, 'Applications of some measures of multivariate skewness and kurtosis in testing normality and robustness studies', Sankya: Ind. J. of Stat., vol. 36, Ser. B, pt. 2, 1974, pp. 142-154.

[71] W.J. Masek and M.S. Paterson, ' A faster algorithm computing string edit distances', J. Comput. and Sys. Sciences, vol. 20, 1980, pp. 18-31.

[72] A. Mercer and A. Rosenfeld, 'An array grammar programming system', Comm. ACM, vol. 16, 1973, pp. 299-305.

[73] N. Metropolis, et al., 'Equations of state calculations by fast computing machines', J. Chem. Physics, vol. 21, 1953, pp. 1087-1091.

[74] D.L. Milgram and A. Rosenfeld, 'Array automata and array grammars', Proc. IFIP Congress-71, vol. 1, 1971, North Holland, pp.69-74.

[75] B. Moayer and K.S. Fu, 'An application of stochastic languages to fingerprint pattern recognition', Pattern Recognition, vol. 8, 1976, pp. 173-179.

[76] R.K. Moore, 'A dynamic programming algorithm for the distances between two finite areas', IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-1, 1979, pp. 86-88.

[77] J.L. Mundy and R.E. Joynson, 'Automatic visual inspection using syntactic analysis', Proc. IEEE Conf. on Pattern Recognition and Image Processing, Troy, New York, 1977, pp. 144-147.

[78] R. Narasimhan, 'Labelling schemata and syntactic description pictures', Info. and Control, vol. 7, 1964, pp. 151-179.

[79] R. Narasimhan, 'Syntax directed intrepretation of classes of pictures', Comm. ACM, vol. 9, 1966, pp. 166-173.

[80] T. Okuda, et al., 'A method for correction of garbled words based on the Levenshtein metric', IEEE Trans. Comput., vol. C-25, 1976, pp. 172-177.

[81] T.W. Pao, 'A solution of the syntactical induction-inference problem for a non-trivial subset of context free languages', Tech. Rep. 69-19, Moore School of Elec. Engg.,Univ. of Pennsylvania, Philadelphia, 1969.

[82] L.F. Pau, 'Semiconductor IC's: integrated testing and algorithms for visual inspection', Proc. 5th. Intl. Conf. on Pattern Recog., Miami Beach, Florida, vol. 1, 1980, pp. 238-240.

[83] T. Pavlidis, Structural Pattern Recognition, Springer-Verlag, Berlin, Heidelburg, 1977.

[84] T. Pavlidis and F. Ali, 'A hierarchical syntactic shape analyzer', IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-1, 1979, pp. 2-9.

[85] E. Persoon and K.S. Fu, 'Shape description using Fourier descriptors', IEEE Trans. Sys. Man and Cyber., vol. SMC-7, 1977, pp. 170-179.

[86] J.L. Pfaltz, 'Web grammars and picture description', Computer Graphics and Image Processing, vol. 1, 1972, pp. 193-220.

[87] G.M. Porter, et al., 'Automatic visual inspection of blind holes in metal surfaces', Proc. IEEE Conf. on Pattern Recognition and Image Processing, Chicago, Illinois, 1979, pp. 83-86.

[88] W.K. Pratt, Digital Image Processing, John Wiley (Interscience), New York, 1978.

[89] C.J. Preston, 'Generalized Gibbs states and Markov random fields', Adv. Appl. Prob., vol. 5, 1973, pp. 242-261.

[90] K. Rao and K. Balck, 'Type classification of fingerprints: a syntactic approach', IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-2, 1980, pp. 223-231.

[91] A. Rosenfeld, 'Isotonic grammars, parallel grammars and picture grammars', Machine Intelligence, vol. 4, 1971, pp. 281-294.

[92] A. Rosenfeld and D.L. Milgram, 'Web automata and web grammars', Machine Intelligence, vol. 7, 1972, pp. 307-324.

[93] A. Rosenfeld, 'Array grammar normal forms', Info. and Control, vol. 23, 1973, pp. 173-182.

[94] G. Rozenberg and A. Salomaa, The Mathematical Theory of L Systems, Academic Press, 1980.

[95] A. Salomaa, Formal Languages, Academic Press, New York, 1973.

[96] C.E. Shannon, 'A mathematical theory of communication', Bell Sys. Tech. Jour., vol. 27, 1948, pp. 379-423.

[97] A.C. Shaw, 'A formal picture description scheme as a basis for picture processing systems', Info. and Control, vol. 14, 1969, pp. 9-52

[98] A.C. Shaw, 'Picture graphs, grammars and parsing', from Frontiers of Pattern Recognition (S. Watanabe, ed.), Academic Press, New York, 1972.

[99] G. Siromoney, et al., 'Abstract families of matrices and picture languages',Computer Graphics and Image Processing, vol. 1, 1972, pp. 284-307.

[100] G. Siromoney, et al., 'Picture Languages with array rewriting rules', Info. and Control, vol. 22, 1973, pp. 447-470.

[101] R. Siromoney and K. Krithivasan, 'Parallel context free languages', Info. and Control, vol. 24, 1974, pp. 155-162.

[102] R. Siromoney and G. Siromoney, 'Parallel OL languages', Intl. J. Comput. Math., vol. 5, Sec. A, 1975, pp. 109-123.

[103] A.R. Smith III, 'Cellular automata and formal languages', IEEE Eleventh Ann. Symp. Switch. and Automata Theory, CA, 1970, pp. 216-224.

[104] A.R. Smith III, 'Cellular automata complexity tradeoffs', Info. and Control, vol. 18, 1971, pp. 466-482.

[105] A.R. Smith III, 'Real time language recognition by one dimensional cellular automata', J. Comput. Sys. Sciences, vol. 6, 1971, pp. 233-253.

[106] S.R. Sternberg, 'Parallel architectures for image processing', Intl. IEEE COMPSAC, Chicago, 1979, pp. 1-6.

[107] G.C. Stockman, et al., 'Structural pattern recognition of cartoid waves using a general waveform parsing system', Comm. ACM, vol. 19, 1976, pp. 688-695.

[108] G.T. Tang and T.S. Huang, 'A syntactic-semantic approach to image understanding and creation', IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-1, 1979, pp. 135-144.

[109] E. Tanaka and Y. Kikuchi, 'A metric between pictures', Tech. Rep., Dept. of Info. Science, Utsunomiya Univ., Japan, 1981.

[110] R.A. Thompson, 'Determination of probabilistic grammars for functionally specified probabilistic-measure languages', IEEE Trans. Comput., vol. C-23, 1973, pp. 603-614.

[111] H. Tropt, 'Analysis by synthesis search for semantic segmentation applied to workpiece recognition', Proc. 5th. Intl. Conf. on Pattern Recog., Miami Beach, Florida, vol. 1, 1980, pp.241-244.

[112] W.H. Tsai and K.S. Fu, 'A syntactic-statistical approach to recognition of industrial objects', Proc. 5th. Intl. Conf. on Pattern Recog., Miami Beach, Florida, vol. 1, 1980, pp. 251-259.

[113] R.M. Wagner and M.J. Fischer, 'The string-to-string correction problem', J. ACM, vol. 21, 1973, pp. 168-173.

[114] P.S. Wang, 'Sequential/parallel matrix array languages', Journal of Cyber., vol. 5, 1975, pp. 19-36.

[115] P.S. Wang, 'Recognition of two dimensional patterns', Tech. Rep. no. CS-TR-8-16, Dept. of Computer Science, Univ. of Oregon, Eugene Oregon, 1978.

[116] P.S. Wang, 'Syntactic structures of parallel isometric array patterns',Tech. Rep. no. CIS-TR-80-1, Dept. of Comp. and Info. Science, Univ. of Oregon, Eugene, Oregon, 1980.

[117] P.S. Wang, 'Some new results on isotonic array grammars', Info. Proc. Letters, vol. 10, 1980, pp. 129-131.

[118] R.M. Wharton, 'Grammatical inference and approximation', Ph.D. Thesis, Univ. of Toronto, Toronto, 1973.

[119] E. Yodokawa and N. Honda, 'On two dimensional pattern generation grammars', Systems, Computers and Controls, vol. 1, 1970, pp. 6-13.

[120] C.T. Zahn and R.Z. Roskies, 'Fourier descriptors for plane closed curves', IEEE Trans. Computers, vol. C-21, 1972, pp. 269-281.

[121] B. Zavidovique and G. Stamon, 'An automated process for electronics scheme analysis', Proc. 5th. Intl. Conf. on Pattern

Recog., Miami Beach, Florida, vol. 1, 1980, pp. 248-250.