Communication and Cybernetics 14

Syntactic Pattern Recognition, Applications

Edited by K.S. Fu

Springer-Verlag Berlin Heidelberg New York

Syntactic Pattern Recognition, Applications

Edited by K. S. Fu

With Contributions by
J. E. Albus R. H. Anderson J. M. Brayer R. DeMori
H.-Y. F. Feng K. S. Fu S. L. Horowitz B. Moayer
T. Pavlidis W. W. Stallings P. H. Swain T. Vámos

With 135 Figures

Springer-Verlag
Berlin Heidelberg New York 1977

Professor KING SUN FU, PhD

Purdue University, School of Electrical Engineering
West Lafayette, IN 47907, USA

Professor Dr. WOLF DIETER KEIDEL

I. Physiologisches Institut der Universität Erlangen-Nürnberg
D-8520 Erlangen, Fed. Rep. of Germany

Professor Dr. HANS WOLTER

Institut für Angewandte Physik der Universität,
D-3550 Marburg/Lahn, Fed. Rep. of Germany

ISBN 3-540-07841-X Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-07841-X Springer-Verlag New York Heidelberg Berlin

Library of Congress Cataloging in Publication Data. Main entry under title: Syntactic pattern recognition. (Communication and cybernetics; 14). Bibliography: p. . Includes index. 1. Pattern perception-Addresses, essays, lectures. 2. Pattern recognition systems-Addresses, essays, lectures. I. Albus, J.E. II. Fu, King Sun, 1930—. Q327.S93. 001.53'4. 76-42196. ISBN 0-387-07841-X

Monophoto typesetting, offset printing, and book binding: Brühlsche Universitätsdruckerei, Giessen

Preface

The many different mathematical techniques used to solve pattern recognition problems may be grouped into two general approaches: the decision-theoretic (or discriminant) approach and the syntactic (or structural) approach. In the decision-theoretic approach, a set of characteristic measurements, called features, are extracted from the patterns. Each pattern is represented by a feature vector, and the recognition of each pattern is usually made by partitioning the feature space. Applications of decision-theoretic approach include character recognition, medical diagnosis, remote sensing, reliability and socio-economics. A relatively new approach is the syntactic approach. In the syntactic approach, each pattern is expressed in terms of a composition of its components. The recognition of a pattern is usually made by analyzing the pattern structure according to a given set of rules. Earlier applications of the syntactic approach include chromosome classification, English character recognition and identification of bubble and spark chamber events. The purpose of this monograph is to provide a summary of the major recent applications of syntactic pattern recognition.

After a brief introduction of syntactic pattern recognition in Chapter 1, the nine main chapters (Chapters 2–10) can be divided into three parts. The first three chapters concern with the analysis of waveforms using syntactic methods. Specific application examples include peak detection and interpretation of electrocardiograms and the recognition of speech patterns. The next five chapters deal with the syntactic recognition of two-dimensional pictorial patterns. Applications examples consist of Chinese character recognition, recognition of geometric figures and two-dimensional mathematical expressions, classification of fingerprint patterns and interpretation of Earth Resources Satellite data. The last chapter treats the problem of recognition of three-dimensional objects, namely, machine parts and industrial objects.

It is the authors of the individual chapters whose contributions made this volume possible. The editor wishes to express his heartfelt appreciation to the authors for their cooperation in its rapid completion.

West Lafayette, Indiana
Winter 1976/77

K. S. Fu

Contributors

ALBUS, JOHN EDWARD
7921 Brimfield, St. Apt. 3 Van Nuys, CA 91402, USA

ANDERSON, ROBERT HELMS
Rand Corporation, Santa Monica, CA 90406, USA

BRAYER, JOHN MARVIN
Department of Electrical Engineering and Computer Sciences, University of New Mexico, Albuquerque, NM 87106, USA

DEMORI, RENATO
Politecnio di Torino, Istituto di Elettrotecnica, Corso Duca degli Abruzzi 24, I-10129 Torino

FENG, HOU-YUAN FRANK
Department of Electrical Engineering, Princeton University, Princeton, NJ 08540, USA

FU, KING SUN
Purdue University, School of Electrical Engineering, West Lafayette, IN 47907, USA

HOROWITZ, STEVEN LESLIE
Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720, USA

MOAYER, BIJAN
The Teheran Polytechnic, Shiraz Avenue, Teheran, Iran

PAVLIDIS, THEODOSIOS
Department of Electrical Engineering, Princeton University, Princeton, NJ 08540, USA

STALLINGS, WILLIAM WALTER
Advanced Computer Techniques Corporation, 1501 Wilson Boulevard, Arlington, VA 22209, USA

SWAIN, PHILIP H.
Purdue University, School of Electrical Engineering, West Lafayette, IN 47907, USA

VÁMOS, TIBOR
Computer and Automation Institute, Hungarian Academy of Sciences, H-1502 Budapest POB 63

Contents

1. Introduction to Syntactic Pattern Recognition

K. S. Fu

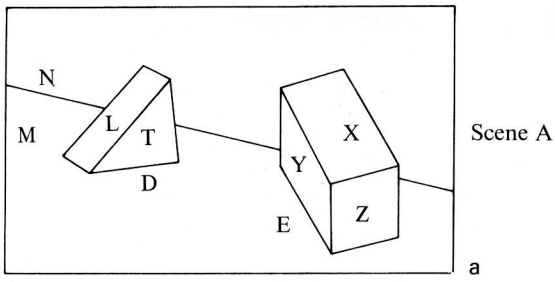
With 10 Figures

1.1 Syntactic (Structural) Approach to Pattern Recognition

Most of the developments in pattern recognition research during the past decade deal with the decision-theoretic approach [1.1–11] and its applications. In some pattern recognition problems, the structural information which describes each pattern is important, and the recognition process includes not only the capability of assigning the pattern to a particular class (to classify it), but also the capacity to describe aspects of the pattern which make it ineligible for assignment to another class. A typical example of this class of recognition problem is picture recognition, or more generally speaking, scene analysis. In this class of recogniton problems, the patterns under consideration are usually quite complex and the number of features required is often very large which makes the idea of describing a complex pattern in terms of a (hierarchical) composition of simpler subpatterns very attractive. Also, when the patterns are complex and the number of possible descriptions is very large, it is impractical to regard each description as defining a class (for example, in fingerprint and face identification problems, recognition of continuous speech, Chinese characters, etc.). Consequently, the requirement of recognition can be satisfied only by a description for each pattern rather than the simple task of classification.

Example 1.1: The pictorial patterns shown in Fig. 1.1a can be described in terms of the hierarchical structures shown in Fig. 1.1b.

In order to represent the hierarchical (tree-like) structural information of each pattern, that is, a pattern described in terms of simpler subpatterns and each simpler subpattern again be described in terms of even simpler subpatterns, etc., the syntactic or structural approach has been proposed [1.12–16]. This approach draws an analogy between the (hierarchical, tree-like) structure of patterns and the syntax of languages. Patterns are specified as building up out of subpatterns in various ways of composition just as phrases and sentences are built up by concatenating words and words are built up by concatenating characters. Evidently, for this approach to be advantageous, the simplest subpatterns selected, called "pattern primitives", should be much easier to recognize than the patterns themselves. The "language" which provides the structural description of patterns in terms of a set of pattern primitives and their composition operations, is sometimes called "pattern description language". The rules governing the composition of primitives into patterns are usually specified by the co-called "grammar" of the pattern description language. After each primitive within the pattern is identified, the recognition



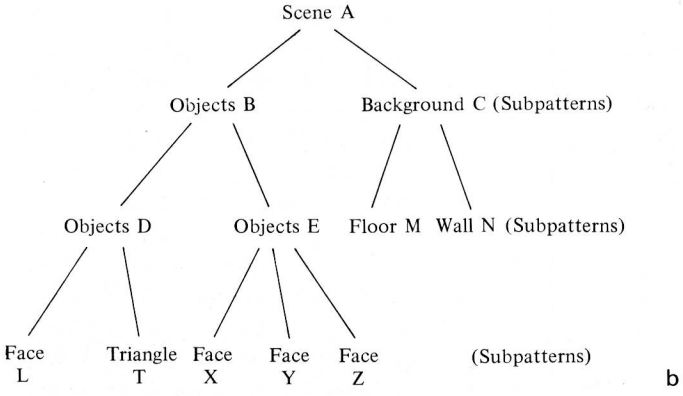


Fig. 1.1a and b. The pictorial pattern *A* and its hierarchical structural descriptions

process is accomplished by performing a syntax analysis or parsing of the "sentence" describing the given pattern to determine whether or not it is syntactically (or grammatically) correct with respect to the specified grammar. In the meantime, the syntax analysis also produces a structural description of the sentence representing the given pattern (usually in the form of a tree structure).

The syntactic approach to pattern recognition provides a capability for describing a large set of complex patterns using small sets of simple pattern primitives and of grammatical rules. The various relations or composition operations defined among subpatterns can usually be expressed in terms of logical and/or mathematical operations. As can be seen later, one of the most attractive aspects of this capability is the use of recursive nature of a grammar. A grammar (rewriting) rule can be applied any number of times, so it is possible to express in a very compact way some basic structural characteristics of an infinite set of sentences. Of course, the practical utility of such an approach depends upon our ability to recognize the simple pattern primitives and their relationships represented by the composition operations.

An alternative representation of the structural information of a pattern is to use a "relational graph". For example, a relational graph of Pattern A in Fig. 1.1a is shown in Fig. 1.2. Since there is a one-to-one corresponding

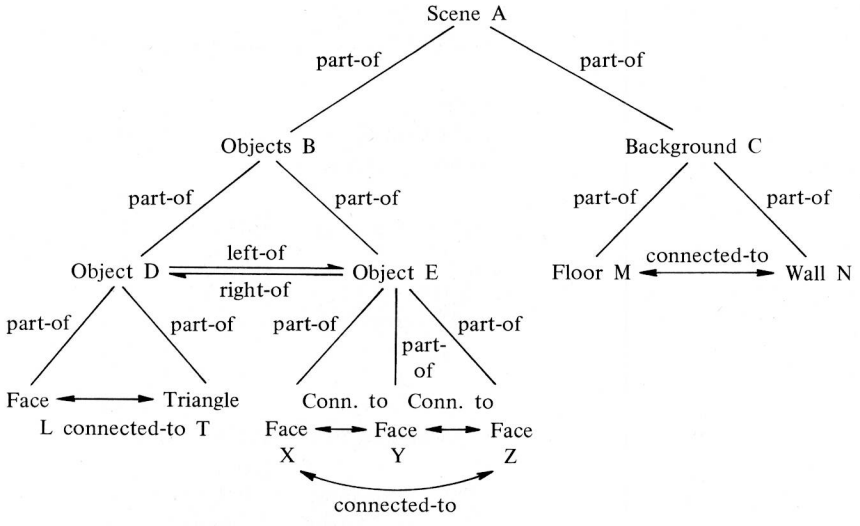


Fig. 1.2. A relational graph of scene *A*

relation between a linear graph and a matrix, a relational graph can certainly also be expressed as a "relational matrix". In using the relational graph for pattern description, we can broaden the class of allowed relations to include any relation that can be conveniently determined from the pattern. With this generalization, we may possibly express richer descriptions than we can with tree structures. However, the use of tree structures does provide us a direct channel to adapt the techniques of formal language theory to the problem of compactly representing and analyzing patterns containing significant structural information.

We briefly introduce some important definitions and notations in this section.

Definition 1.1: A (phrase-structure) grammar G is a four-triple

$$G=(V_N, V_T, P, S)$$

where

V_N is a finite set of nonterminals,

V_T is a fintie set of terminals,

$S \in V_N$ is the start symbol,

and P is a finite set of rewrite rules or productions denoted by

$$\alpha \rightarrow \beta . \tag{1.1}$$

α and β are strings over the union of V_N and V_T and with α involving at least one symbol of V_N.

The following notations are frequently used.

1) V^* is the set of all strings of symbols in V, including λ, the string of length 0, $V^+ = V^* - \{\lambda\}$.

2) If x is a string, x^n is x written n times.

3) $|x|$ is the length of the string x, or the number of symbols in string x.

4) $\eta \underset{G}{\Rightarrow} \gamma$, or a string η directly generates or derives a string γ if $\eta = \omega_1 \alpha \omega_2$, $\gamma = \omega_1 \beta \omega_2$, and $\alpha \rightarrow \beta$ is a production in P.

5) $\eta \overset{*}{\underset{G}{\Rightarrow}} \gamma$, or a string η generates or derives a string γ if there exists a sequence of strings $\zeta_1, \zeta_2, \ldots \zeta_n$ such that $\eta = \zeta_1$, $\gamma = \zeta_n$, $\zeta_i \Rightarrow \zeta_{i+1}$, $i = 1, 2, \ldots, n-1$. The sequence of strings $\zeta_1, \zeta_2, \ldots, \zeta_n$ is called a derivation of γ from η.

Definition 1.2: The language generated by grammar G is

$$L(G) = \{x | x \in V_T^* \text{ and } S \overset{*}{\underset{G}{\Rightarrow}} x\} . \tag{1.2}$$

That is, the language consists of all strings or sentences of terminals generated from the start symbol S.

Definition 1.3: In (1.1) if $|\alpha| \leqq |\beta|$, the grammar is called a type 1 or context-sensitive grammar. If $\alpha = A \in V_N$, the grammar is called a type 2 or context-free grammar. If, in addition to $\alpha = A$, $\beta = aB$ or $\beta = a$, where $a \in V_T$ and $B \in V_N$, the grammar is called a type 3, or finite-state, or regular grammar.

The languages generated by context-sensitive, context-free, and finite-state (regular) grammars are called context-sensitive, context-free, and finite-state (regular) languages, respectively.

Example 1.2: Consider the context-free grammar

$$G = (V_N, V_T, P, S)$$

where $V_N = \{S, A, B\}$, $V_T = \{a, b\}$, and P[1]:

1) $S \rightarrow aB$
2) $S \rightarrow bA$
3) $A \rightarrow aS$
4) $A \rightarrow bAA$
5) $A \rightarrow a$
6) $B \rightarrow bS$
7) $B \rightarrow aBB$
8) $B \rightarrow b$.

The language generated by G, $L(G)$, is the set of all sentences or strings in V_T^+ consisting of an equal number of a's and b's. Typical generations or derivations of sentences include

$$S \overset{(1)}{\Rightarrow} aB \overset{(8)}{\Rightarrow} ab$$
$$S \overset{(1)}{\Rightarrow} aB \overset{(6)}{\Rightarrow} abS \overset{(2)}{\Rightarrow} abbA \overset{(5)}{\Rightarrow} abba$$
$$S \overset{(2)}{\Rightarrow} bA \overset{(4)}{\Rightarrow} bbAA \overset{(4)}{\Rightarrow} bbbAAA \overset{(5)}{\Rightarrow} bbbaAA \overset{(5)}{\Rightarrow} bbbaaA \overset{(5)}{\Rightarrow} bbbaaa$$

where the parenthesized number indicates the production used.

[1] For convenience, we can also used the shorthand notation $S \rightarrow aB|bA$ for representing productions 1) and 2). Similarly, we can use $A \rightarrow aS|bAA|a$ for productions 3), 4), and 5), and use $B \rightarrow bS|aBB|b$ for productions 6), 7), and 8).

An alternative method for describing any derivation in a context-free grammar is the use of derivation or parse trees. A derivation tree for a context-free grammar can be constructed according to the following procedure:

1) Every node of the tree has a label, which is a symbol in V_N or V_T.

2) The root of the tree has the label S.

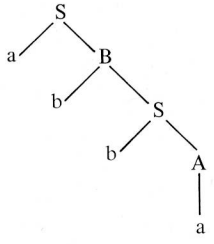
3) If a node has at least one descendant other than itself, and has the label A, then $A \in V_N$.

4) If nodes $n_1, n_2, \ldots n_k$ are the direct descendants of node n (with label A) in the order from left to right, with labels $A_1, A_2, \ldots, A_k$, respectively, then

$$A \rightarrow A_1 A_2, \ldots, A_k$$

must be a production in P.

For example, the derivation $S \overset{*}{\Rightarrow} abba$ in Example 1.2 can be described by the following derivation tree:



1.2 Syntactic Pattern Recognition System

A syntactic pattern recognition system can be considered as consisting of three major parts; namely, preprocessing, pattern description or representation, and syntax analysis[2]. A simple block diagram of the system is show in Fig. 1.3. The functions of preprocessing include i) pattern encoding and approximation, and ii) filtering, restoration and enhancement. An input pattern is first coded or approximated by some convenient form for further processing. For example, a black-and-white picture can be coded in terms of a grid (or a matrix) of 0's and 1's, or a waveform can be approximated by its time samples or a truncated Fourier series expansion. In order to make the processing in the later stages of the system more efficient, some sort of "data compression" is often applied at this stage. Then, techniques of filtering, restoration and/or enhancement will be used to clean the noise, to restore the degradation, and/or to improve the quality of the coded (or approximated) patterns. At the output of the preprocessor, presumably, we have patterns with reasonably "good quality". Each preprocessed pattern is then represented by a language-like structure (for example, a string or a graph). The operation of this pattern-

2 The division of three parts is for convenience rather than necessity. Usually, the term "linguistic pattern recognition" refers primarily to the pattern representation (or description) and the syntax analysis.

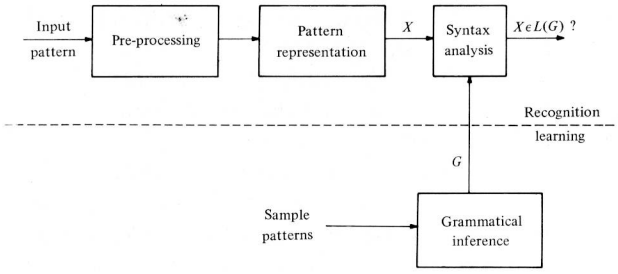


Fig. 1.3. Block diagram of a syntactic pattern recognition system

representation process consists of i) pattern segmentation, and ii) primitive (feature) extraction. In order to represent a pattern in terms of its subpatterns, we must segmentize the pattern and, in the meantime, identify (or extract) the primitives and relations in it. In other words, each preprocessed pattern is segmentized into subpatterns and pattern primitives based on prespecified syntactic or composition operations; and, in turn, each subpattern is identified with a given set of pattern primitives. Each pattern is now represented by a set of primitives with specified syntactic operations. For example, in terms of "concatenation" operation, each pattern is represented by a string of (concatenated) primitives. More sophisticated systems should also be able to detect various syntactic relations within the pattern. The decision on whether or not the representation (pattern) is syntactically correct (i.e., belongs to the class of patterns described by the given syntax or grammar) will be performed by the "syntax analyzer" or "parser". When performing the syntax analysis or parsing, the analyzer can usually produce a complete syntactic description, in terms of a parse or parsing tree, of the pattern provided it is syntactically correct. Otherwise, the pattern is either rejected or analyzed on the basis of other given grammars, which presumably describe other possible classes of patterns under consideration.

Conceptually, the simplest form of recognition is probably "template-matching". The string of primitives representing an input pattern is matched against strings of primitives representing each prototype or reference pattern. Based on a selected "matching" or "similarity" criterion, the input pattern is classified in the same class as the prototype pattern which is the "best" to match the input. The hierarchical structural information is essentially ignored. A complete parsing of the string representing an input pattern, on the other hand, explores the complete hierarchical structural description of the pattern. In between, there are a number of intermediate approaches. For example, a series of tests can be designed to test the occurrences or non-occurrence of certain subpatterns (or primitives) or certain combinations of subpatterns or primitives. The result of the tests (for example, through a table look-up, a decision tree, or a logical operation) is used for a classification decision. Notice that each test may be a template-matching scheme or a parsing for a subtree representing a subpattern. The selection of an appropriate approach for recognition usually depends upon the problem requirement. If a complete pattern description is required for recognition, parsing is necessary. Otherwise, a

complete parsing could be avoided by using other simpler approaches to improve the efficiency of the recognition process.

In order to have a grammar describing the structural information about the class of patterns under study, a grammatical inference machine is required which can infer a grammar from a given set of training patterns in language-like representations[3]. This is analogous to the "learning" process in a decision-theoretic pattern recognition system [1.1–11, 17–20]. The structural description of the class of patterns under study is learned from the actual sample patterns from that class. The learned description, in the form of a grammar, is then used for pattern description and syntax analysis (see Fig. 1.3). A more general form of learning might include the capability of learning the best set of primitives and the corresponding structural description for the class of patterns concerned.

1.3 Selection of Pattern Primitives

As was discussed in Section 1.1, the first step in formulating a linguistic model for pattern description is the determination of a set of primitives in terms of which the patterns of interest may be described. This will be largely influenced by the nature of the data, the specific application in question, and the technology available for implementing the system. There is no general solution for the primitive selection problem at this time. The following requirements usually serve as a guideline for selecting pattern primitives.

i) The primitives should serve as basic pattern elements to provide a compact but adequate description of the data in terms of the specified structural relations (e.g., the concatenation relation).

ii) The primitives should be easily extracted or recognized by existing non-linguistic methods, since they are considered to be simple and compact patterns and their structural information not important.

For example, for speech patterns, phonemes are naturally considered as a "good" set of primitives with the concatenation relation[4]. Similarly, strokes have been suggested as primitives in describing handwriting. However, for general pictorial patterns, there is no such "universal picture element" analogous to phonemes in speech or strokes in handwriting[5]. Sometimes, in order to provide an adequate description of the patterns, the primitives should contain the information which is important to the specific application in question. For example, If the size (or shape or location) is important in the recognition problem, then the primitives should contain information relating to size (or shape or location) so that patterns from different classes are distinguishable by whatever method

[3] At present, this part is performed primarily by the designer.

[4] The view of continuous speech as composed of one sound segment for each successive phoneme is, of course, a simplification of facts.

[5] It is also interesting to see that the extraction of phonemes in continuous speech and that of strokes in handwriting is not a very easy task with respect to the requirement ii) specified above.

is to be applied to analyze the descriptions. This requirement often results in a need for semantic information in describing primitives [1.12].

Requirement ii) may sometimes conflict with requirement i) due to the fact that the primitives selected according to requirement i) may not be easy to recognize using existing techniques. On the other hand, requirement ii) could allow the selection of quite complex primitives as long as they can be recognized. With more complex primitives, simpler structural descriptions (e.g., simple grammars) of the patterns could be used. This tradeoff may become quite important in the implementation of the recognition system. An example is the recognition of two-dimensional mathematical expressions in which characters and mathematical notations are primitives. However, if we consider the characters as subpatterns and describe them in terms of simpler primitives (e.g., strokes or line segments), the structural descriptions of mathematical expressions would be more complex than the case of using characters directly as primitives.

One of the earliest papers describing the decomposition of pictorial patterns into primitives [1.20a] presented a conceptually appealing method which allows the recognition system to (heuristically) determine the primitives by inspection of training samples. A pattern is first examined by a programmed scan. The result of the scan is to produce descriptions of segments of the picture (subpictures) which are divisions conveniently produced by the scanning process, and not necessarily true divisions. The scanning process also includes pre-processing routines for noise-cleaning, gap-filling, and curve-following. The subpictures obtained in the scan are analyzed and connected, when appropriate, into true picture parts; a description is given in terms of the length and slope of straight-line segments and the length and curvature of curved segments. The structural relations among various segments (primitives) of a picture are expressed in terms of a connection table (Table of Joins). The assembly program produces a "statement" which gives a complete description of the pattern. The description is independent of the orientation and the size of the picture, the lengths of the various parts being given relative to one another. It is, in effect, a coded representation of the pattern and may be regarded as a one-dimensional string consisting of symbols chosen from a specified alphabet. The coded representation gives the length, slope and curvature of each primitive, together with details of the ends and joins to other primitives. No explicit consideration is given to formalizing the pattern syntax.

A formal model for the abstract description of English cursive script has been proposed by EDEN and HALLE [1.21]. The primitives are four distinct line segments in the form of a triple

$$\sigma_j = [(x_{j_1}, y_{j_1}), (x_{j_2}, y_{j_2}), \theta_j] \tag{1.3}$$

where (x_j, y_j)'s represent the approximate location of the end points of the line segment, and θ_j refers to the sense of rotation from the first to the second end point. θ_j is positive if the sense of rotation is clockwise and negative